Information and Computation 208 (2010) 1368–1397

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Subsequential transducers: a coalgebraic perspective ☆

Helle Hvid Hansen [1]

Formal Methods Group, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Available online 13 May 2010

Keywords:
Subsequential transducer
Word function
Coalgebra
Normalisation
Differential

ABSTRACT

Subsequential transducers combine (input) language recognition with transduction and thereby generalise classic deterministic automata as well as Mealy and Moore type state machines. These well known subclasses all have a natural coalgebraic characterisation, and the question arises whether their coalgebraic modelling can be extended to subsequential transducers and their underlying structures. In this paper, we show that although subsequential structures cannot generally be regarded as coalgebras, the subclass of normalised structures do form a subcategory of coalgebras. Moreover, normalised structures are reflective in the category of all subsequential structures, and a final normalised structure exists. The existence and properties of the minimal subsequential transducer can be derived from this result. We also show that for the class of subsequential structures in which all states are accepting, an alternative coalgebraic representation is obtained by taking differentials. This differential representation gives rise to a new method of deciding equivalence and computing minimal representations which does not involve normalisation. Both normalisation and taking differentials can be formalised as functors into reflective subcategories of coalgebras, and we can therefore see these constructions as coalgebraisation.

1. Introduction

In this paper we study subsequential transducers from a coalgebraic perspective. Subsequential transducers can be described as deterministic automata which produce output words on transitions and terminal output at accepting states. A subsequential transducer realises a partial word function, called its behaviour, by its transformation of accepted input words to output words. This combination of language recognition and transduction makes subsequential transducers useful in areas such as lexical analysis, coding theory, computer arithmetic (cf. [13]) and more recently, in speech and language processing (cf. [21,29]).

Subsequential transducers were introduced by Schützenberger [40] as a generalisation of sequential transducers [12]. They have since been studied, in particular, by Choffrut [9,10] who showed that subsequential transducers can be minimised via a normalisation construction; that any partial word function is realised by a minimal subsequential transducer, and that the Ginsburg–Rose characterisation theorem for sequential functions can be generalised to the subsequential case. Other results may be found in [4,5,8].

In the abovementioned work, subsequential transducers were studied from the algebraic perspective on automata and formal languages (cf. [11,12]) where syntax, congruence and initiality are fundamental notions. Coalgebra provides a dual perspective based on general notions of behaviour, bisimilarity and finality. Many different types of state-based systems have been identified as coalgebras (cf. [3,15,25,37]), but automata form the archetypical examples of coalgebras (over Set).

☆ This paper is the extended journal version of the work first published in the CMCS 2008 conference as: "H.H. Hansen, Coalgebraising subsequential transducers, Proceedings of the Ninth Workshop on Coalgebraic Methods in Computer Science (CMCS 2008), ENTCS 203 (5) 2008, Pp 109-129, doi:10.1016/j.entcs.2008.05.022.

[1] Partially supported by NWO grant 612.000.316.

E-mail address: h.h.hansen@tue.nl

0890-5401/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2009.10.008

In particular, classic deterministic automata and Mealy/Moore machines can all be modelled coalgebraically such that their traditional semantics coincides with the coalgebraic semantics (cf. [35,37]). These well known automata classes form natural subclasses of subsequential transducers.

Our motivation for carrying out coalgebraic modelling is that by identifying a class of automata as coalgebras, we obtain general results regarding bisimulation, minimisation and structural theory (cf. [37]). We also gain a larger mathematical perspective in which to view existing results and constructions. Moreover, having a coalgebraic modelling opens up the possibility of applying general results and techniques for specifying properties of coalgebras using logic languages, see for example [7,26,31]. In this paper we focus on the basic parts of the coalgebraic modelling, and leave logic and specification as future work.

The main aim of this paper is to place subsequential transducers and their underlying structures in a coalgebraic framework. In particular, Choffrut's results on the existence of a minimal subsequential transducer [10] strongly suggest that a final object exists at some level. On the other hand, for someone familiar with coalgebra it is not difficult to see that the word function semantics of subsequential transducers is not captured by the associated coalgebraic notion of behaviour. We would like to obtain a clear picture of the mathematical underpinnings of these observations. This paper has three main parts.

In the first part (Section 3), we define subsequential transducers, structures and morphisms. We show that the subcategories of coaccessible, normalised and minimal subsequential structures form a sequence of nested, full, reflective subcategories of the category of all subsequential structures. Each move to a reflective subcategory can be seen as a step towards an optimal representation. These results give a clear break-down of the essential steps of minimisation and parallel known results for deterministic automata (cf. [1,2]).

In the second part (Section 4), we turn to the coalgebraic modelling. Although subsequential structures are easily seen to have the type of coalgebras for a functor $\mathcal{S}$, the associated notion of $\mathcal{S}$-coalgebra morphism does not include all subsequential morphisms. Hence the category Subseq of subsequential structures and morphisms is not coalgebraic. However, we show that the subcategory $\mathsf{NSubseq}$ of normalised subsequential structures is a full subcategory of $\mathsf{Coalg}(\mathcal{S})$. This result is the basis for our slogan that *normalisation is coalgebraisation*. Moreover, we prove that $\mathsf{NSubseq}$ has a final object Ω, and since $\mathsf{NSubseq}$ is reflective in Subseq, Ω is also final in Subseq. Finally, we describe how to adapt the minimisation algorithm of DFAs to normalised structures.

In the third part (Section 5), we present an alternative coalgebraic modelling for subsequential transducers in which all states are final. We call such subsequential transducers (and their underlying structures) *step-by-step*. This coalgebraic representation is obtained by a structural transformation which corresponds to taking the differential of the behaviour. These so-called differential representations can be seen as sequential transducers which produce output in the free group $B^{(*)}$ rather than the free monoid B^*, and they can be modelled as coalgebras for a functor $\mathcal{S}_0^{(*)}$. We can therefore also say that *taking differentials is coalgebraisation*. Moreover, like normalisation, taking differentials is functorial and a reflector. The practical interest of this coalgebraic characterisation is that it provides us with an alternative method for deciding equivalence of step-by-step transducers which does not require normalisation.

This paper is an extended and improved version of [18], and its main contributions are: (1) The classification of various subclasses of subsequential structures and transducers in terms of reflective subcategories and their coalgebraic or non-coalgebraic nature. In particular, the observation that normalisation and taking differentials are reflectors to categories of coalgebras. (2) The identification of a final subsequential structure which explains existing results on minimal subsequential transducers from [10]. (3) A new approach for deciding equivalence of step-by-step transducers.

2. Preliminaries

2.1. Functions and words

Let X and Y be sets. A (partial) function from X to Y is denoted by $f : X \dashrightarrow Y$. We will write $f : X \to Y$ when f is a total function from X to Y. The domain and range of $f : X \dashrightarrow Y$ are denoted $\mathrm{dom}(f)$ and $\mathrm{ran}(f)$, respectively. For a function $f : X \dashrightarrow Y$, and subsets $C \subseteq X$ and $D \subseteq Y$, the f-image of C is denoted $f(C) = \{f(x) \in Y \mid x \in C \cap \mathrm{dom}(f)\}$, the inverse f-image of D is $f^{-1}(D) = \{x \in \mathrm{dom}(f) \mid f(x) \in D\}$, and the restriction of f to C is $f|_C$.

The free monoid over a set X is the monoid $(X^*, \varepsilon, \mathbin{.})$ where X^* is the set of all words over X, ε is the empty word, and $u \mathbin{.} w$, or simply uw, denotes the concatenation of two words $u, w \in X^*$. For all $u, w \in X^*$, we write $u \preceq w$ if u is a prefix of w, i.e., there exists a $v \in X^*$ such that $w = uv$. The length of a word w is denoted by $|w|$. If $f, g : X \to B^*$, then $f \mathbin{.} g : X \to B^*$ is the function defined by $(f \mathbin{.} g)(x) = f(x) \mathbin{.} g(x)$. The free group over X is denoted by $X^{(*)}$, and the formal inverse of x in $X^{(*)}$ is written $\overline{x}$. For $w \in X^*$, the inverse of $w = x_1 x_2 \ldots x_k$ is $\overline{w} = \overline{x_k} \ldots \overline{x_2}\,\overline{x_1}$, and $\overline{\varepsilon} = \varepsilon$. The symbol $\mathbin{.}$ also denotes multiplication in $X^{(*)}$. Hence, if $w = uv \in X^*$, then $\overline{u} \mathbin{.} w = v$ and $w \mathbin{.} \overline{v} = u$. An example where u is not a prefix of $w \in X^*$, is given by $\overline{aab} \mathbin{.} ab = \overline{ab} \mathbin{.} b$. A subset $T \subseteq X^*$ is called *prefix-closed* if whenever $u \preceq w$ and $w \in T$ then $u \in T$. A partial function $f : X^* \dashrightarrow Y^*$ is *prefix-preserving* if $\mathrm{dom}(f)$ is prefix-closed, and for all $u, w \in \mathrm{dom}(f)$, if $u \preceq w$ then $f(u) \preceq f(w)$. For a set $S \subseteq X^*$ of words, we denote by $lcp(S)$ the longest common prefix of words in S with the convention that $lcp(\emptyset)$ is undefined.

2.2. Coalgebra

Some previous familiarity with coalgebra will be useful to the reader, but we will provide all the basic coalgebraic definitions relevant for this paper. For a more detailed introduction, including many examples, we refer to Rutten [37]. We do, however, assume the reader is familiar with basic notions such as category and functor, and the constructions of product, coproduct and exponentiation. In general, for an object C in a category C, we denote the identity C-morphism on C by id_C.

Our coalgebras will be based on Set, the category of sets and total functions. Given a functor $\mathcal{F}: \mathsf{Set} \to \mathsf{Set}$, an *$\mathcal{F}$-coalgebra* is a pair $\mathbb{S} = (S, \sigma)$ where S is a set and $\sigma: S \to \mathcal{F}(S)$ is a function, often called an $\mathcal{F}$-coalgebra structure on S. A function $f: S_1 \to S_2$ is an *$\mathcal{F}$-coalgebra morphism* from (S_1, σ_1) to (S_2, σ_2) if $\mathcal{F}(f) \circ \sigma_1 = \sigma_2 \circ f$, that is, if the following diagram commutes:

$$\begin{array}{ccc} S_1 & \xrightarrow{f} & S_2 \\ {\scriptstyle \sigma_1}\downarrow & & \downarrow{\scriptstyle \sigma_2} \\ \mathcal{F}(S_1) & \xrightarrow{\mathcal{F}(f)} & \mathcal{F}(S_2) \end{array}$$

$\mathcal{F}$-coalgebras and $\mathcal{F}$-coalgebra morphisms form a category which we denote by $\mathsf{Coalg}(\mathcal{F})$.

Let (S, σ) be an $\mathcal{F}$-coalgebra. An $\mathcal{F}$-coalgebra (S', σ') is a *subcoalgebra* of (S, σ), if $S' \subseteq S$ and the inclusion map $\iota: S' \to S$ is an $\mathcal{F}$-coalgebra morphism. Subcoalgebras are determined by their carrier ([37, Proposition 6.1]), meaning that if (S', σ') and (S', σ'') are subcoalgebras of (S, σ), then $\sigma' = \sigma''$. We can therefore think of the subcoalgebras of (S, σ) as certain subsets of S. If the intersection of all subcoalgebras containing a state $s \in S$ is again a subcoalgebra of $\mathbb{S} = (S, \sigma)$, then we call it the *subcoalgebra generated by s in* $\mathbb{S}$ and denote it by $\langle s \rangle_{\mathbb{S}}$. We will also use the fact that if f is an $\mathcal{F}$-coalgebra morphism from (S_1, σ_1) to (S_2, σ_2), then the image $f(S_1)$ is a subcoalgebra of (S_2, σ_2). If f is injective, then (S_1, σ_1) is isomorphic to $f(S_2)$, since any bijective $\mathcal{S}$-coalgebra morphism is an isomorphism in $\mathsf{Coalg}(\mathcal{S})$. We refer to [37, Proposition 2.3, Theorem 6.3] for these last two facts.

Given an $\mathcal{F}$-coalgebra $\mathbb{S} = (S, \sigma)$ a relation $R \subseteq S \times S$ is a *congruence on* $\mathbb{S}$, if R is the kernel of some $\mathcal{F}$-coalgebra morphism f from (S, σ) to some (S', σ'). The identity relation on S is a congruence, and congruences are closed under unions. Consequently, a largest congruence on $\mathbb{S}$ always exists (cf. [17]). A congruence R is an equivalence relation and we denote the set of R-equivalence classes on S by S/R. If R is a congruence, then there exists a unique $\mathcal{F}$-coalgebra structure $\sigma_R: S/R \to \mathcal{F}(S/R)$ such that the quotient map $\rho_R: S \to S/R$ is an $\mathcal{F}$-coalgebra morphism from (S, σ) to $(S/R, \sigma_R)$ (see e.g. [37, Propositions 5.7 and 5.8]). We denote the quotient structure $(S/R, \sigma_R)$ by $\mathbb{S}/R$. An $\mathcal{F}$-coalgebra $\mathbb{S}$ is called *minimal* if it has no proper quotients, that is, if the largest congruence on $\mathbb{S}$ is the identity relation.

Coalgebras come with abstract notions of behaviour and bisimulation. Let two $\mathcal{F}$-coalgebras $\mathbb{S}_1 = (S_1, \sigma_1)$ and $\mathbb{S}_2 = (S_2, \sigma_2)$ be given. Two states $s_1 \in S_1$ and $s_2 \in S_2$ are called *behaviourally equivalent* if there exist an $\mathcal{F}$-coalgebra (S, σ) and $\mathcal{F}$-coalgebra morphisms f_j from (S_j, σ_j) to (S, σ), $j \in \{1, 2\}$, such that $f_1(s_1) = f_2(s_2)$, in which case we write $s_1 \equiv_{\mathcal{F}} s_2$. On an $\mathcal{F}$-coalgebra $\mathbb{S}$, the behavioural equivalence relation $\equiv_{\mathcal{F}}$ is the largest congruence on $\mathbb{S}$, see e.g. [17]. One way of proving that two states are behaviourally equivalent is to show that they are linked by a bisimulation. A relation $Z \subseteq S_1 \times S_2$ is an *$\mathcal{F}$-bisimulation between $\mathbb{S}_1$ and $\mathbb{S}_2$*, if Z can be equipped with coalgebraic structure $\zeta: Z \to \mathcal{F}(Z)$ such that the projections $\pi_i: Z \to S_i$, $i \in \{1, 2\}$, are $\mathcal{F}$-coalgebra morphisms. Two states $s_1 \in S_1$ and $s_2 \in S_2$ are *$\mathcal{F}$-bisimilar* (notation: $s_1 \sim_{\mathcal{F}} s_2$) if there exists an $\mathcal{F}$-bisimulation Z between $\mathbb{S}_1$ and $\mathbb{S}_2$ such that $\langle s_1, s_2 \rangle \in Z$. It is always the case that $s_1 \sim_{\mathcal{F}} s_2$ implies $s_1 \equiv_{\mathcal{F}} s_2$, however, the converse holds only under certain conditions on $\mathcal{F}$ (cf. [37, Theorem 4.3]).

An object Ω in a category C is a *final object* in C, if for any C-object C there is a unique C-morphism h_C from C to Ω called the *final C-morphism (from C)*. Note that $h_\Omega = id_\Omega$. A *final $\mathcal{F}$-coalgebra* is a final object in $\mathsf{Coalg}(\mathcal{F})$. In general, a final $\mathcal{F}$-coalgebra need not exist, but if it does, it is unique up to isomorphism, and we will therefore sometimes speak of *the* final coalgebra. The elements of a final $\mathcal{F}$-coalgebra can be thought of as all possible behaviours of $\mathcal{F}$-coalgebras. The following theorem summarises some fundamental properties of final $\mathcal{F}$-coalgebras. The proof can be found in [37, Section 9].

Theorem 2.1. *Let $\Omega = (S_\Omega, \sigma_\Omega)$ be a final $\mathcal{F}$-coalgebra.*

1. *For all $s, t \in S_\Omega$: $s \sim_{\mathcal{F}} t$ iff $s = t$.*
2. *All subcoalgebras of Ω are minimal.*

Example 2.2. Many known structures are identified as being coalgebras (see e.g. [37]). We mention in particular two types of automata which are special instances of subsequential transducers. The first is classical deterministic automata over an alphabet A. A classic deterministic automaton can be seen as a map $\langle o, d \rangle: Q \to \mathbf{2} \times Q^A$, where $d: Q \to Q^A$ is the next-state function and the output function $o: Q \to \mathbf{2} = \{0, 1\}$ defines whether a state $q \in Q$ is accepting ($o(q) = 1$) or not ($o(q) = 0$). Such maps are coalgebras for the functor $\mathcal{A}ut(X) = \mathbf{2} \times X^A$, and it is straightforward to show that $\mathcal{A}ut$-coalgebra morphisms coincide with the well known morphisms of deterministic automata; that the final $\mathcal{A}ut$-coalgebra consists of the set of all languages $\mathcal{P}(A^*)$, and the final $\mathcal{A}ut$-coalgebra morphism is the map that sends a state q to the set of words accepted from q (see e.g. [35]).

The second example is given by Mealy machines (cf. [28]). A Mealy machine with input in A and output in B is a coalgebra of the type $t: Q \to (B \times Q)^A$ (cf. [37]), and a final Mealy-coalgebra is obtained by equipping the set $\{f: A^+ \to B\}$ of functions which map non-empty words over A to B with the following structure: $f \mapsto \lambda a.\langle f(a), f_a\rangle$ where $f_a(w) = \overline{f(a)} \cdot f(aw)$ for all $w \in A^+$.

In this paper all functors considered are polynomial, i.e., they are constructed from constant sets, identity, product, coproduct and exponentiation. We will make use of the following facts.

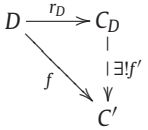
Proposition 2.3. *Let $\mathcal{F}: \mathsf{Set} \to \mathsf{Set}$ be a polynomial functor.*

1. *The final $\mathcal{F}$-coalgebra exists.*
2. *For all $\mathcal{F}$-coalgebras (S, σ) and all $s, t \in S$: $s \sim_{\mathcal{F}} t$ iff $s \equiv_{\mathcal{F}} t$.*
3. *For all $\mathcal{F}$-coalgebras (S, σ), the bisimilarity relation $\sim_{\mathcal{F}}$ is the largest congruence on $\mathbb{S}$, hence $\mathbb{S}/\sim_{\mathcal{F}}$ is a minimal $\mathcal{F}$-coalgebra.*
4. *For all $\mathcal{F}$-coalgebras $\mathbb{S} = (S, \sigma)$ and all $s \in S$, the generated subcoalgebra $\langle s\rangle_{\mathbb{S}}$ exists.*
5. *For all $\mathcal{F}$-coalgebras $\mathbb{S}_1$ and $\mathbb{S}_2$ containing states s_1 and s_2, respectively: $s_1 \sim_{\mathcal{F}} s_2$ iff $\kappa_1(s_1) \sim_{\mathcal{F}} \kappa_2(s_2)$, where $\kappa_i : \mathbb{S}_i \to \mathbb{S}_1 + \mathbb{S}_2$, $i \in \{1, 2\}$, are the coproduct injections.*

Proof. These properties follow from the fact that polynomial functors are standard, continuous and preserve weak pullbacks. Details may be found in [37]. □

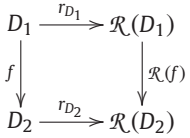
2.3. Reflective subcategories

We recall the definition and some facts of reflective subcategories (see e.g. [1,27]). Let C be a subcategory of D, and D an object in D. A C-*reflection arrow* for D is a D-morphism $r_D: D \to C_D$ to some C-object C_D which has the following universal property. For any $C' \in \mathsf{C}$ and any D-morphism $f: D \to C'$ there is a unique C-morphism $f': C_D \to C'$ such that $f = f' \circ r_D$. That is, the following diagram commutes.



The subcategory C of D is *reflective in* D if for every D-object D there is a C-reflection arrow for D. As an example, we mention that in the category of deterministic automata (DAs) the subcategory of minimal DAs is reflective (see [2, Chapter VI.1] and also [1, Example, 4.17]).

An equivalent formulation of reflective subcategory is the following. A subcategory C of D is *reflective in* D if the embedding functor $\mathcal{E}: \mathsf{C} \to \mathsf{D}$ has a left adjoint $\mathcal{R}: \mathsf{D} \to \mathsf{C}$. This left adjoint $\mathcal{R}$ is called a *reflector*. Once a choice of reflection arrow has been made for every D-object D, the functor $\mathcal{R}$ can be defined by $\mathcal{R}(D) = C_D$, and for $f: D_1 \to D_2$, $\mathcal{R}(f)$ is the C-arrow determined by the following diagram:



This implies that if C is reflective in D, then a final object C_0 in C is also final in D, due to the bijection of Hom-sets: $\mathsf{D}(D, C_0) \cong \mathsf{C}(\mathcal{R}(D), C_0)$. Namely, the unique morphism $h_{\mathcal{R}(D)}: \mathcal{R}(D) \to C_0$ corresponds to a unique morphism $h_D: D \to C_0$. Finally, since reflectors are adjoints, and adjoints compose, if A is a reflective subcategory of B, and B is a reflective subcategory of C, then A is a reflective subcategory of C.

3. Subsequential structures and transducers

In this section we review the basic definitions of subsequential transducers (cf. [9,10]), and define subsequential structures and their morphisms. We characterise the full subcategories of coaccessible, normalised and minimal subsequential structures. In particular, we show that they form a sequence of nested, reflective subcategories in the category of all subsequential structures.

3.1. Basic definitions

Throughout this paper, we assume we are given two (possibly infinite) sets A and B, which we refer to as the input and output alphabet, respectively. A subsequential transducer can be seen as a deterministic automaton which for every accepted

input word from A^* produces an output word in B^*. If the input word leads to a non-accepting state, then the output will be considered undefined. The transition structure is assumed deterministic, but not total, that is, for each state and each input letter there is at most one transition available. The output is generated by outputting words on transitions and a terminal output word at final states. Moreover, the subsequential transducer may be equipped with an initial prefix which is a word that will be prefixed to the output generated from processing the input.

In coalgebra, we usually focus on structures without a designated initial state, and this will also be the case here. The underlying structure of a subsequential transducer is obtained by leaving out its initial state and initial prefix.

Definition 3.1 (*subsequential structure and transducer*)**.** A *subsequential structure* is a 4-tuple $\mathbb{S} = (Q, o, d, r)$ where Q is a set of states, $o\colon Q \to (A \dashrightarrow B^*)$ is an *output function*, $d\colon Q \to (A \dashrightarrow Q)$ is a *next-state function* and $r\colon Q \dashrightarrow B^*$ is a *terminal output function*. We require that for all $q \in Q$, $\mathrm{dom}(o(q)) = \mathrm{dom}(d(q)) =: \mathrm{supp}(q)$, called the support of q. The set of *final (or accepting)* states of $\mathbb{S}$ is $F := \mathrm{dom}(r)$. If $q \notin F$ then q is called an *internal* state. The *underlying deterministic automaton* of $\mathbb{S}$ is the structure (Q, d, F). If $Q = \emptyset$, then $\mathbb{S}$ is called the empty subsequential structure.

A *subsequential transducer* is a 6-tuple $\mathbb{T} = (Q, o, d, r, i, m)$ where (Q, o, d, r) is a subsequential structure, and if $Q \neq \emptyset$, $i \in Q$ is the *initial state*, and $m \in B^*$ is the *initial prefix*. In case $Q = \emptyset$, i and m are considered undefined, and $\mathbb{T}$ is called the empty transducer.

The reason we allow the empty subsequential structure/transducer is that later (cf. Definition 3.13) we want the operation of taking the coaccessible part to be a map on subsequential structures, and this operation can result in an empty set of states.

A path in a subsequential structure $\mathbb{S} = (Q, o, d, r)$ starting in state q_0 and ending in state q_n is a sequence $(q_0, a_0, w_0), (q_1, a_1, w_1), \ldots, (q_{n-1}, a_{n-1}, w_{n-1}), q_n$ where $n \geq 0$, $q_0, \ldots, q_n$ are states in Q, $a_0, \ldots, a_{n-1}$ are elements of A and $w_0, \ldots, w_{n-1}$ are words in B^* such that for all $j < n$, $d(q_j)(a_j) = q_{j+1}$ and $o(q_j)(a_j) = w_j$. The parameter n is called the length of the path, the words $a_0a_1 \ldots a_{n-1}$ and $w_0w_1 \ldots w_{n-1}$ are called the input and output labels of the path, respectively. A path is called *final* if it ends in a final state, and a state q is *coaccessible* if there exists a final path starting in q. The set of coaccessible states of a subsequential structure $\mathbb{S}$ (or transducer $\mathbb{T}$) will be denoted by $Coacc(\mathbb{S})$ (respectively, $Coacc(\mathbb{T})$). In a subsequential transducer, a state q is *accessible* (or reachable) if there is a path from the initial state to q. The set of accessible states of a subsequential transducer $\mathbb{T}$ are denoted by $Acc(\mathbb{T})$. A subsequential transducer is called *trimmed*, if all its states are accessible and coaccessible.

Let $\mathbb{S} = (Q, o, d, r)$ be a subsequential structure and $q \in Q$ a state. We extend the output and next-state functions at q to maps $o(q)\colon A^* \dashrightarrow B^*$ and $d(q)\colon A^* \dashrightarrow Q$ in the following standard manner. For a state $q \in Q$, $a \in A$ and $w \in A^*$, we define

$$d(q)(\varepsilon) = q, \qquad d(q)(wa) = d(d(q)(w))(a),$$
$$o(q)(\varepsilon) = \varepsilon, \qquad o(q)(wa) = o(q)(w) \cdot o(d(q)(w))(a).$$

with the proviso that the left side is defined only if the right side is. The set of words accepted from q in the underlying DA (Q, d, F) is called the *input language of* q and we denote it by $\mathcal{L}(q)$, i.e.,

$$\mathcal{L}(q) = \{w \in A^* \mid d(q)(w) \in F\}. \tag{1}$$

A subsequential transducer $\mathbb{T}$ realises a partial word function by its transformation of input words to output words. Similarly, given a subsequential structure $\mathbb{S}$ and a state q in $\mathbb{S}$, we can consider the partial word function realised by $\mathbb{S}$ when starting in q.

Definition 3.2 (*behaviour*)**.** Given a subsequential structure $\mathbb{S} = (Q, o, d, r)$ and a state $q \in Q$, the *behaviour of* q *(in* $\mathbb{S}$*)* is the partial function $[\![q]\!]_{\mathbb{S}}\colon A^* \dashrightarrow B^*$ defined for all $w \in \mathcal{L}(q)$ by:

$$[\![q]\!]_{\mathbb{S}}(w) = o(q)(w) \cdot r(d(q)(w)). \tag{2}$$

Given two subsequential structures $\mathbb{S}$ and $\mathbb{S}'$, two states q in $\mathbb{S}$ and q' in $\mathbb{S}'$ are *equivalent* if $[\![q]\!]_{\mathbb{S}} = [\![q']\!]_{\mathbb{S}'}$.

The *behaviour* of a subsequential transducer $\mathbb{T} = (\mathbb{S}, i, m)$ is the partial function $[\![\mathbb{T}]\!]\colon A^* \dashrightarrow B^*$ defined for all $w \in \mathcal{L}(i)$ by:

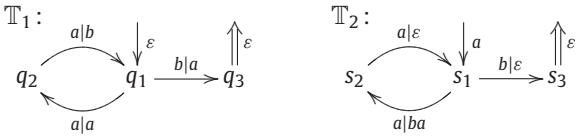
$$[\![\mathbb{T}]\!](w) = m \cdot [\![i]\!]_{\mathbb{S}}(w). \tag{3}$$

We say that $\mathbb{T}$ realises $[\![\mathbb{T}]\!]$, and two subsequential transducers $\mathbb{T}_1$ and $\mathbb{T}_2$ are *equivalent* if $[\![\mathbb{T}_1]\!] = [\![\mathbb{T}_2]\!]$.

For notational simplicity we sometimes leave out the subscript from $[\![q]\!]_{\mathbb{S}}$ when $\mathbb{S}$ is clear from the context, or we use some appropriate indexing, for example, $[\![q]\!]_1$ instead of $[\![q]\!]_{\mathbb{S}_1}$.

Example 3.3. Consider the subsequential transducers $\mathbb{T}_1$ and $\mathbb{T}_2$ depicted below. The initial state is marked by an incoming, sourceless arrow labelled with the initial prefix; a transition from a state q to a state q' on input letter a with output w is

illustrated as an arrow from q to q' with label $a|w$; and final states are marked with an outgoing double-arrow labelled with the terminal output.



It is not difficult to see that $\mathbb{T}_1$ and $\mathbb{T}_2$ compute the same partial function $f\colon \{a,b\}^* \dashrightarrow \{a,b\}^*$, where $\mathrm{dom}(f) = \{(aa)^k b \mid k \in \omega\}$, and for all $k \in \omega, f((aa)^k b) = (ab)^k a$. Hence $\mathbb{T}_1$ and $\mathbb{T}_2$ are equivalent.

Choffrut introduced in [10] a notion of morphism between trimmed subsequential transducers. This definition is based on his observation in [9] that it is possible to systematically shift some of the output letters "upstream" without changing the input–output behaviour. For example, in Example 3.3 above, the two transducers realise the same function, and the underlying DAs are isomorphic. The only difference is that internally $\mathbb{T}_2$ produces its output a bit faster than $\mathbb{T}_1$. Informally, Choffrut's definition of morphism is a map between states which respects transitions of the underlying DAs together with the requirement that an output shift exists which makes the two subsequential transducers produce their output in a synchronised manner.

We wish to define a notion of morphism for arbitrary subsequential transducers and in particular for subsequential structures. We do so by a slight variation on Choffrut's definition. See Remark 3.12 at the end of this section for details on how the two relate to each other.[2] Since we view subsequential structures as more fundamental than subsequential transducers, we will first define morphisms between subsequential structures, and then add conditions for the transducer case. We note that the conditions (next) and (init) in Definition 3.4 below should be read as saying that the left side is defined if and only if the right side is, and if both are defined the two must be equal.

Definition 3.4 (*subsequential morphism*)**.** Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be two subsequential structures. A function $\alpha\colon Q_1 \dashrightarrow Q_2$ is a *subsequential morphism* from $\mathbb{S}_1$ to $\mathbb{S}_2$ (notation: $\alpha\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$), if there exists a function $\beta\colon Q_1 \to B^*$ such that the following conditions are satisfied for all $q \in Q_1$:

(next) $\forall a \in A\colon \alpha(d_1(q)(a)) = d_2(\alpha(q))(a)$,

(out) $\forall a \in A\colon$ if $q, d_1(q)(a) \in \mathrm{dom}(\alpha)$
then $\beta(q) \cdot o_2(\alpha(q))(a) = o_1(q)(a) \cdot \beta(d_1(q)(a))$,

(acc) $\alpha^{-1}(F_2) = F_1$,

(term-out) if $q \in F_1$ then $\beta(q) \cdot r_2(\alpha(q)) = r_1(q)$.

Given two subsequential transducers $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_1)$ and $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$, a *subsequential (transducer) morphism* from $\mathbb{T}_1$ to $\mathbb{T}_2$ is a subsequential morphism $\alpha\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ such that α and its witnessing function $\beta\colon Q_1 \to B^*$ satisfy:

(init) $\alpha(i_1) = i_2$,

(ε-in) if $i_1 \in \mathrm{dom}(\alpha)$ then $m_2 = m_1 \cdot \beta(i_1)$.

Remark 3.5. We will use the notation $(\alpha, \beta)\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ to say that α is a subsequential morphism from $\mathbb{S}_1$ to $\mathbb{S}_2$ with witnessing function β.

Condition (next) can be illustrated by the diagrams in Fig. 1. The diagrams should be read as follows. The left side of (next) is defined iff the solid arrows of the (next-left) diagram are given. Consequently, the dotted arrows should exist such that the resulting diagram commutes. Similarly, for the diagram (next-right).

One might have expected that in the (next-left) diagram the upper arrow with label α should have been solid rather than the lower one. The next-state condition corresponding to this variation is the condition required of *proper state mappings* of partial DAs in [12, III.4], and it ensures that in transducers, α must be defined on all accessible states. Definition 3.4, on the other hand, ensures that α must be defined on all coaccessible states (Lemma 3.8). We find the (next-left) condition appropriate since we view subsequential structures as potential subsequential transducers, namely, each state in a subsequential structure has the potential to become the initial state of a subsequential transducer. A morphism of subsequential structures should therefore include all states that have the potential to become a transducer with non-trivial behaviour, i.e., all coaccessible states.

[2] We remark that our current definition of subsequential morphisms differs from the one given earlier in [18], which turns out to be the correct notion for coaccessible structures, but not for subsequential structures in general.

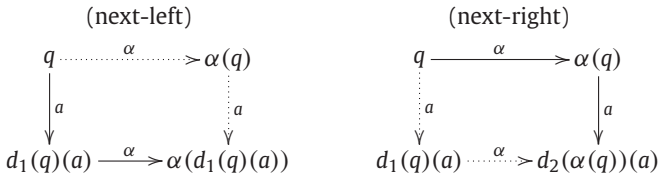


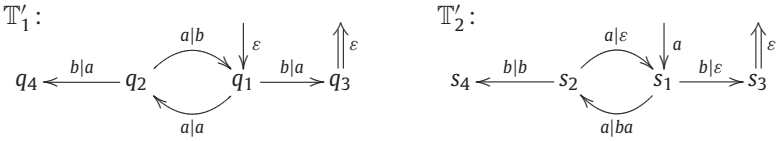
Fig. 1. Diagrammatic description of condition (next).

Example 3.6. We can now verify that in Example 3.3 the map $\alpha(q_j) = s_j, j \in \{1, 2, 3\}$, is a subsequential morphism from $\mathbb{T}_1$ to $\mathbb{T}_2$ by taking $\beta(q_1) = a$, $\beta(q_2) = ba$ and $\beta(q_3) = \varepsilon$. For example, to see that the (out) condition holds, we have

$$\begin{aligned}
\beta(q_1) \cdot o_2(s_1)(a) &= a \cdot ba = a \cdot ba = o_1(q_1)(a) \cdot \beta(q_2),\\
\beta(q_2) \cdot o_2(s_2)(a) &= ba \cdot \varepsilon = b \cdot a = o_1(q_2)(a) \cdot \beta(q_1),\\
\beta(q_1) \cdot o_2(s_1)(b) &= a \cdot \varepsilon = a \cdot \varepsilon = o_1(q_1)(b) \cdot \beta(q_3).
\end{aligned}$$

We allow a subsequential morphism α to be partial in order to be able to ignore states that have trivial behaviour. This is the reason why the condition (out) is restricted to states in $\mathrm{dom}(\alpha)$. We illustrate with an example.

Example 3.7. Consider the two variations on the subsequential transducers of Example 3.3.



It can easily be confirmed that the (now partial) map $\alpha(q_j) = s_j, j \in \{1, 2, 3\}$, is a subsequential morphism from $\mathbb{T}'_1$ to $\mathbb{T}'_2$ by defining β as in Example 3.6 for q_1, q_2, q_3, and letting $\beta(q_4)$ be any value in B^*. The states q_4 and s_4 are not coaccessible, and hence do not contribute to the behaviour of $\mathbb{T}'_1$ and $\mathbb{T}'_2$. In Lemma 3.15 below, we will see that the witnessing function β of any subsequential morphism is uniquely defined on all coaccessible states. In this example, this implies that q_4 cannot be in the domain of any subsequential morphism from $\mathbb{T}'_1$ to $\mathbb{T}'_2$, since there is no value for $\beta(q_4) \in B^*$ which would satisfy the (out) condition.

We now show that states with trivial behaviour are the only states that are allowed to be ignored by subsequential morphisms, and that they respect input languages.

Lemma 3.8. *Let $\mathbb{S}_1$ and $\mathbb{S}_2$ be subsequential structures and $\alpha : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ a subsequential morphism.*

1. *$Coacc(\mathbb{S}_1) \subseteq \mathrm{dom}(\alpha)$.*
2. *For all $q \in \mathrm{dom}(\alpha)$: $\mathcal{L}(q) = \mathcal{L}(\alpha(q))$.*
3. *For all $q \in \mathrm{dom}(\alpha)$: $q \in Coacc(\mathbb{S}_1)$ iff $\alpha(q) \in Coacc(\mathbb{S}_2)$.*

Proof. Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$, and assume $\alpha : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$. Item (1): By definition, $q \in Coacc(\mathbb{S}_1)$ iff there exists a $w \in A^*$ such that $d_1(q)(w) \in F_1$. We show by induction on the length of w that for all $q \in Q_1$, $d_1(q)(w) \in F_1$ implies $q \in \mathrm{dom}(\alpha)$. Base case: If $d_1(q)(\varepsilon) = q \in F_1$, then by (acc) we have $\alpha(q) \in F_2$, hence $q \in \mathrm{dom}(\alpha)$. The induction step follows easily from (next-left) and the induction hypothesis.

Item (2): We prove by induction on the length of $w \in A^*$ that for all $q \in \mathrm{dom}(\alpha)$: $d_1(q)(w) \in F_1$ iff $d_2(\alpha(q))(w) \in F_2$. Item (2) then follows immediately. The base case follows from (acc). For the induction step, let $a \in A$ and $v \in A^*$. We have:

$$d_1(d_1(q)(a))(v) \in F_1 \overset{\text{(IH)}}{\iff} d_2(\alpha(d_1(q)(a)))(v) \in F_2 \overset{\text{(next)}}{\iff} d_2(d_2(\alpha(q))(a))(v) \in F_2$$

and hence $d_1(q)(av) \in F_1$ iff $d_2(\alpha(q))(av) \in F_2$.

Item (3) follows from item (2), since for all $q \in Q_1$, $q \in Coacc(\mathbb{S}_1)$ iff $\mathcal{L}(q) \neq \emptyset$, similarly for $\alpha(q)$. □

Subsequential morphisms preserve the behaviour of subsequential transducers, but not necessarily the behaviour of states, i.e., $\alpha : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ *does not* imply that for all q in $\mathrm{dom}(\alpha)$, $[\![q]\!]_1 = [\![\alpha(q)]\!]_2$. This can be observed in Example 3.3.

Instead, given a subsequential morphism α with witnessing function β, the state behaviour of $\alpha(q)$ can be obtained from the state behaviour of q by explicit mention of $\beta(q)$.

Proposition 3.9. *Let $\mathbb{T} = (\mathbb{S}_1, i_1, m_1)$ and $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ be subsequential transducers. We have:*

1. *If $(\alpha, \beta) : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$, then for all $q \in \mathrm{dom}(\alpha)$: $[\![q]\!]_1 = \beta(q) \cdot [\![\alpha(q)]\!]_2$.*
2. *If $\alpha : \mathbb{T}_1 \dashrightarrow \mathbb{T}_2$, then $[\![\mathbb{T}_1]\!] = [\![\mathbb{T}_2]\!]$.*

Proof. Item (1): Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$, $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ and assume that $(\alpha, \beta) : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$. From Lemma 3.8(2) it follows immediately that for all $q \in \mathrm{dom}(\alpha)$, $\mathrm{dom}([\![q]\!]_1) = \mathrm{dom}([\![\alpha(q)]\!]_2)$, and, in particular, $[\![q]\!]_1$ is the empty map if and only if $[\![\alpha(q)]\!]_2$ is the empty map. We prove by induction on the length of $w \in \mathrm{dom}([\![q]\!]_1)$ that $[\![q]\!]_1(w) = \beta(q) \cdot [\![\alpha(q)]\!]_2(w)$. Base case:

$$[\![q]\!]_1(\varepsilon) = r_1(q) \overset{(\text{term}-\text{out})}{=} \beta(q) \cdot r_2(\alpha(q)) = \beta(q) \cdot [\![\alpha(q)]\!]_2(\varepsilon).$$

Induction step: Let $w = av \in \mathrm{dom}([\![q]\!]_1)$ where $a \in A$ and $v \in A^*$. Note that this implies that q and $d(q)(a)$ are in $\mathrm{dom}(\alpha)$. We have:

$$\begin{aligned}
[\![q]\!]_1(av) &= o_1(q)(a) \cdot [\![d_1(q)(a)]\!]_1(v) \\
&\overset{(\text{IH})}{=} o_1(q)(a) \cdot \beta(d_1(q)(a)) \cdot [\![\alpha(d_1(q)(a))]\!]_2(v) \\
&\overset{(\text{out})}{=} \beta(q) \cdot o_2(\alpha(q))(a) \cdot [\![\alpha(d_1(q)(a))]\!]_2(v) \\
&\overset{(\text{next})}{=} \beta(q) \cdot o_2(\alpha(q))(a) \cdot [\![d_2(\alpha(q))(a)]\!]_2(v) \\
&= \beta(q) \cdot [\![\alpha(q)]\!](av).
\end{aligned}$$

Item (2): Let $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_2)$, $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ and assume $(\alpha, \beta) : \mathbb{T}_1 \dashrightarrow \mathbb{T}_2$. First consider the case where $i_1 \notin \mathrm{dom}(\alpha)$. Then (init) implies that i_2 is undefined and hence $\mathbb{T}_2$ is empty, and by Lemma 3.8(1), $i_1 \notin \mathit{Coacc}(\mathbb{T}_1)$. Hence $[\![\mathbb{T}_1]\!]$ and $[\![\mathbb{T}_2]\!]$ are both the empty map. Now assume that $i_1 \in \mathrm{dom}(\alpha)$. By definition, for $w \in A^*$: $[\![\mathbb{T}_1]\!](w) = m_1 \cdot [\![i_1]\!]_1(w)$. From item (1) and (init), we get $[\![\mathbb{T}_1]\!](w) = m_1 \cdot \beta(i_1) \cdot [\![i_2]\!]_2(w)$, and finally from ($\varepsilon$-in), $[\![\mathbb{T}_1]\!](w) = m_2 \cdot [\![i_2]\!]_2(w) = [\![\mathbb{T}_2]\!](w)$. □

For a subsequential structure $\mathbb{S} = (Q, o, d, r)$, we define the identity morphism $id_{\mathbb{S}}$ on $\mathbb{S}$ to be the identity map id_Q on the state set Q. It is easily seen that id_Q is a subsequential morphism from $\mathbb{S}$ to $\mathbb{S}$ by taking $\beta = \varepsilon$ (the constant function equal to ε everywhere). Similarly, for a subsequential transducer $\mathbb{T}$ with state set Q, the identity morphism on $\mathbb{T}$ is defined as $id_{\mathbb{T}} := id_Q$. The next lemma shows that subsequential morphisms can be composed.

Lemma 3.10. *Let $\mathbb{S}_j$, $j \in \{1, 2, 3\}$, be subsequential structures, and let $\mathbb{T}_j$, $j \in \{1, 2, 3\}$, be subsequential transducers.*

1. *If $\alpha_1 : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ and $\alpha_2 : \mathbb{S}_2 \dashrightarrow \mathbb{S}_3$ then $\alpha_2 \circ \alpha_1 : \mathbb{S}_1 \dashrightarrow \mathbb{S}_3$.*
2. *If $\alpha_1 : \mathbb{T}_1 \dashrightarrow \mathbb{T}_2$ and $\alpha_2 : \mathbb{T}_2 \dashrightarrow \mathbb{T}_3$ then $\alpha_2 \circ \alpha_1 : \mathbb{S}_1 \dashrightarrow \mathbb{S}_3$.*

Proof. Item (1): Assume $(\alpha_1, \beta_1) : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ and $(\alpha_2, \beta_2) : \mathbb{S}_2 \dashrightarrow \mathbb{S}_3$. Let $\beta : Q_1 \to B^*$ be defined by $\beta(q) = \beta_1(q) \cdot \beta_2(\alpha_1(q))$ if $q \in \mathrm{dom}(\alpha)$ and $\beta(q) = \beta_1(q)$ if $q \in Q_1 \backslash \mathrm{dom}(\alpha_1)$. It is straightforward to check that $(\alpha_2 \circ \alpha_1, \beta) : \mathbb{S}_1 \dashrightarrow \mathbb{S}_3$. Item (2) can easily be verified using item (1). □

Hence subsequential structures and subsequential morphisms form a category Subseq, and subsequential transducers and subsequential transducer morphisms form a category SubseqTra. Note that although subsequential morphisms allow a non-trivial output shift β, isomorphisms in Subseq do not.

Lemma 3.11. *Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be subsequential structures. A subsequential morphism $\alpha : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ is an isomorphism iff α is a bijection and its witnessing function β is constant equal to ε on all coaccessible states.*

Proof. Suppose $(\alpha_1, \beta_1) : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ and $(\alpha_2, \beta_2) : \mathbb{S}_2 \dashrightarrow \mathbb{S}_1$ such that $\alpha_2 \circ \alpha_1 = id_{\mathbb{S}_1}$ and $\alpha_1 \circ \alpha_2 = id_{\mathbb{S}_2}$. Clearly, $\alpha_1 : Q_1 \to Q_2$ is a bijection. Recall that the free group inverse of a word $w \in B^*$ is denoted $\overline{w}$. From condition (term-out) we get that for all $q \in F_1$: $\beta_1(q) = r_1(q) \cdot \overline{r_2(\alpha_1(q))}$ and $\beta_2(\alpha_1(q)) = r_2(\alpha_1(q)) \cdot \overline{r_1(q)}$. This means that $\beta_1(q) = \overline{\beta_2(\alpha(q))}$. Since β_1 and β_2 take values in B^*, we must have that $\beta_1(q) = \beta_2(\alpha_1(q)) = \varepsilon$ for all $q \in F_1$. Using condition (out) and induction on the distance of a state q to F_1, we can extend this argument to show that $\beta_1(q) = \beta_2(\alpha_1(q)) = \varepsilon$ for all $q \in \mathit{Coacc}(\mathbb{S}_1)$. □

Many interesting results on subsequential transducers, subsequential functions[3] and their relationship with rational functions can be found in [5,6,8–10,32], including a characterisation of subsequential functions [9] which generalises the Ginsburg–Rose theorem for sequential functions, and methods of determinisation [5].

Remark 3.12. Definition 3.4 is adapted from Choffrut's definition in [10] of morphisms of trimmed subsequential transducers. We make a few remarks on the differences: (i) Choffrut allows β to take values in $B^* \cup \overline{B^*}$, where $\overline{B^*} = \{\overline{w} \mid w \in B^*\} \subseteq B^{(*)}$. This slightly more general definition allows morphisms to exist from $\mathbb{T}_1$ to $\mathbb{T}_2$, also if $\mathbb{T}_2$ sometimes produces its output slower than $\mathbb{T}_1$. We find this an unnecessary generalisation, since it is not needed to prove the existence of a minimal subsequential transducer (cf. Corollary 4.13). (ii) Choffrut also defines a subsequential morphism as a partial map α, however, since all states in a trimmed subsequential transducer are coaccessible, it follows from conditions (acc) and (next) (cf. Lemma 3.8(i)) that α must be a total function. (iii) Choffrut explicitly includes the witnessing function β in his definition of morphisms of subsequential transducers, i.e., his subsequential morphisms are pairs. We will see in Lemma 3.15 that a subsequential morphism α between trimmed transducers, has exactly one witness β, hence β is implicitly given by α.

3.2. Coaccessible structures

We call a subsequential structure (or transducer) coaccessible, if all its states are coaccessible. We denote by CSubseq the full subcategory of Subseq consisting of coaccessible subsequential structures; similarly, CSubseqTra is the full subcategory of SubseqTra consisting of coaccessible subsequential transducers.

It is well known that given a finite (partial) deterministic automaton, one can obtain the coaccessible part by computing the states that are backwards reachable from the final states, and the same, of course, holds for subsequential structures, since coaccessibility is a property defined by the underlying DA.

Definition 3.13. Let $\mathbb{S} = (Q, o, d, r)$ be a subsequential structure. We define $\mathcal{C}(\mathbb{S}) := (Q', o', d', r)$ where $Q' = \mathit{Coacc}(\mathbb{S})$ and o' and d' are the restrictions of o and d to Q', i.e., for all $q \in Q'$ and $a \in A$, $o'(q)(a) = o(q)(a)$ and $d'(q)(a) = d(q)(a)$, if $d(q)(a) \in Q'$, otherwise $o'(q)(a)$ and $d'(q)(a)$ are undefined.

The following lemma is clear from the definition.

Lemma 3.14. *If $\mathbb{S}$ is a subsequential structure, then $\mathcal{C}(\mathbb{S})$ is a coaccessible subsequential structure.*

We now show that Definition 3.4 ensures that the witnessing output shift function β is uniquely defined on coaccessible states.

Lemma 3.15. *Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be subsequential structures and $\alpha : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ a subsequential morphism. If $\beta : Q_1 \to B^*$ and $\beta' : Q_1 \to B^*$ are both witnessing functions for α, then $\beta\restriction_{\mathit{Coacc}(\mathbb{S}_1)} = \beta'\restriction_{\mathit{Coacc}(\mathbb{S}_1)}$.*

Proof. We show that for all $q \in Q_1$ and all $w \in A^*$, if $d_1(q)(w) \in F_1$ then $\beta(q) = \beta'(q)$. The proof is by induction on the length of w. In the base case, i.e., $q \in F_1$, it follows from (term-out) that $\beta(q) = \beta'(q)$. For the induction step, assume that $a \in A$ and $v \in A^*$ such that $d_1(q)(av) \in F_1$, and let $q_a = d_1(q)(a)$. By induction hypothesis (IH) and (out) it follows that

$$\beta(q) \overset{\text{(out)}}{=} o_1(q)(a) \cdot \beta(q_a) \cdot \overline{o_2(\alpha(q))(a)} \overset{\text{(IH)}}{=} o_1(q)(a) \cdot \beta'(q_a) \cdot \overline{o_2(\alpha(q))(a)} \overset{\text{(out)}}{=} \beta'(q). \quad \square$$

In the previous section, we observed in Lemma 3.8(1) that a subsequential morphism must be defined on all coaccessible states, hence the morphisms in CSubseq are total maps. Together with Lemma 3.15, this leads us to the following characterisation of subsequential morphisms between coaccessible structures.

Proposition 3.16. *Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be coaccessible subsequential structures. A function $\alpha : Q_1 \dashrightarrow Q_2$ is a subsequential morphism from $\mathbb{S}_1$ to $\mathbb{S}_2$ if and only if $\mathrm{dom}(\alpha) = Q_1$ and there exists a unique function $\beta : Q_1 \to B^*$ such that the following conditions are satisfied for all $q \in Q_1$:*

(supp)	$\mathrm{supp}(q) = \mathrm{supp}(\alpha(q))$,
$(\mathrm{next})_C$	$\forall a \in \mathrm{supp}(q) : \alpha(d_1(q)(a)) = d_2(\alpha(q))(a)$,
$(\mathrm{out})_C$	$\forall a \in \mathrm{supp}(q) : \beta(q) \cdot o_2(\alpha(q))(a) = o_1(q)(a) \cdot \beta(d_1(q)(a))$,
(acc)	$\alpha^{-1}(F_2) = F_1$,
(term-out)	*if $q \in F_1$ then* $\beta(q) \cdot r_2(\alpha(q)) = r_1(q)$.

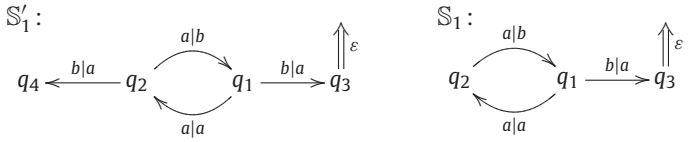
[3] A function $f : A^* \dashrightarrow B^*$ is called *subsequential*, if f is the behaviour of some finite subsequential transducer $\mathbb{T}$, i.e., $\mathbb{T}$ has only finitely many states and transitions.

Proof. First assume that α is a subsequential morphism with witnessing function β. By Lemma 3.8(1), α is total, and hence condition (next) ensures that (supp) and $(\text{next})_C$ holds. The totality of α and (out) together imply that $(\text{out})_C$ holds for α and β. The uniqueness of β follows from Lemma 3.15. Now, assume that α is a total function from Q_1 to Q_2 satisfying the conditions of the proposition with unique witness β. Condition (supp) now ensures that in (next) the left side is defined iff the right side is, and when both are defined, $(\text{next})_C$ ensures that they are equal. To see that (out) holds, note that since α is total, $d(q)(a) \in \text{dom}(\alpha)$ iff $a \in \text{supp}(q)$. Hence (out) follows from $(\text{out})_C$. □

We will use the notation $\alpha : \mathbb{S}_1 \to \mathbb{S}_2$ rather than $\alpha : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$, when $\mathbb{S}_1$ is coaccessible, in order to emphasise that α is a total map.

Next, we will show that CSubseq is reflective in Subseq, by showing that the identity on coaccessible states is a reflection arrow. We first illustrate with a small example that the identity on coaccessible states is a subsequential morphism.

Example 3.17. Consider the subsequential structures $\mathbb{S}_1$ and $\mathbb{S}'_1$ underlying $\mathbb{T}_1$ from Example 3.3 and $\mathbb{T}'_1$ from Example 3.7, as illustrated below.



Clearly, $\mathbb{S}_1 = \mathcal{C}(\mathbb{S}'_1)$, and it is easy to see that $id_{Coacc(\mathbb{S}'_1)} : \mathbb{S}'_1 \dashrightarrow \mathbb{S}_1$ in Subseq with witnessing function β constant equal to ε.

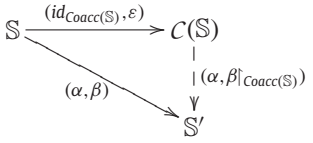
The following theorem confirms that our definition of subsequential morphisms is the correct one. It is also an argument for saying that the right way of thinking about subsequential structures is in terms of their coaccessible part. We will see in the next section that this statement can be sharpened by considering normalised subsequential structures.

Theorem 3.18. *Let $\mathbb{S}$ be a subsequential structure. We have:*

$id_{Coacc(\mathbb{S})} : \mathbb{S} \dashrightarrow \mathcal{C}(\mathbb{S})$ *is a* CSubseq*-reflection arrow for* $\mathbb{S}$.

It follows that CSubseq *is a reflective subcategory of* Subseq, *and the map $\mathcal{C}$ is a functor $\mathcal{C}$*: Subseq $\to$ CSubseq *by defining* $\mathcal{C}(\alpha) = \alpha{\restriction}_{Coacc(\mathbb{S}_1)}$ *for* $\alpha : \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$.

Proof. Let $\mathbb{S}$ be a subsequential structure. It is straightforward to check that $(id_{Coacc(\mathbb{S})}, \varepsilon) : \mathbb{S} \dashrightarrow \mathcal{C}(\mathbb{S})$. It remains to show that $id_{Coacc(\mathbb{S})}$ has the desired universal property, that is, if $(\alpha, \beta) : \mathbb{S} \dashrightarrow \mathbb{S}'$ in Subseq where $\mathbb{S}'$ is in CSubseq, then there is a unique morphism $\alpha' : \mathcal{C}(\mathbb{S}) \to \mathbb{S}'$ in CSubseq such that $\alpha = \alpha' \circ id_{Coacc(\mathbb{S})}$. We claim that $\alpha' = \alpha$ with witnessing function $\beta' = \beta{\restriction}_{Coacc(\mathbb{S})}$ as illustrated in the following diagram.



To prove our claim, we just have to note that by Lemma 3.8(1) and the assumption that $\mathbb{S}'$ is coaccessible, it follows that $\text{dom}(\alpha) = Coacc(\mathbb{S})$, hence $\alpha = \alpha{\restriction}_{Coacc(\mathbb{S})} = \alpha \circ id_{Coacc(\mathbb{S})}$, and α is clearly unique. It is also easy to see that $\beta' = \beta{\restriction}_{Coacc(\mathbb{S})}$ witnesses the fact that $\alpha : \mathcal{C}(\mathbb{S}) \to \mathbb{S}'$ is a morphism. Moreover, β' is unique due to Lemma 3.15. □

3.3. Trimmed transducers

Subsequential transducers are often assumed to be trimmed (cf. [10]), that is, all states are accessible and coaccessible. We now look closer at the operations involved in *trimming* a transducer. First we note that for subsequential transducers, taking the coaccessible part is not always a well defined operation, since doing so on a $\mathbb{T}$ in which the initial state is not coaccessible and the set of final states is not empty will result in an object where the initial state is not an element of the (non-empty) state set. Such objects are not subsequential transducers by our definition. Still we remark that the category CSubseqTra is well defined, only $\mathcal{C}$ is not a functor from SubseqTra to CSubseqTra. Before we can take the coaccessible part, we must first make $\mathbb{T}$ accessible.

Definition 3.19. Let $\mathbb{T} = (Q, o, d, r, i, m)$ be a subsequential transducer. We define $\mathcal{A}(\mathbb{T}) = (Acc(\mathbb{T}), o', d', r', i, m)$ where o', d' and r' are the restrictions of o, d and r to $Acc(\mathbb{T})$. $\mathcal{A}(\mathbb{T})$ is called the *accessible part* of $\mathbb{T}$, and $\mathbb{T}$ is called *accessible* if

$\mathbb{T} = \mathcal{A}(\mathbb{T})$. If $\mathbb{T} = (\mathbb{S}, i, m)$ is an accessible subsequential transducer, then we define $\mathcal{C}(\mathbb{T}) = (\mathcal{C}(\mathbb{S}), i, m)$ with i and m undefined if $Coacc(\mathbb{S}) = \emptyset$.

Trimming a subsequential transducer can be achieved by first taking the accessible part and then the coaccessible part. In particular, note that if $\mathbb{T}$ is accessible, then all states are coaccessible iff the initial state is coaccessible.

Lemma 3.20. *If $\mathbb{T}$ is a subsequential transducer, then $\mathcal{A}(\mathbb{T})$ is an accessible subsequential transducer and $\mathcal{C}(\mathcal{A}(\mathbb{T}))$ is a trimmed subsequential transducer.*

Let ASubseqTra and TSubseqTra denote the full subcategories of SubseqTra consisting of accessible and trimmed subsequential transducers, respectively.

The notions of accessibility and coaccessibility are properties of the underlying deterministic automaton, and the following proposition can easily be adapted to a statement about (partial) deterministic automata.

Proposition 3.21. *We have:*

1. *For all $\mathbb{T} \in$ SubseqTra, $id_{Acc(\mathbb{T})} : \mathcal{A}(\mathbb{T}) \dashrightarrow \mathbb{T}$ is a subsequential morphism.*
2. *For all $\mathbb{T} \in$ ASubseqTra, $id_{Coacc(\mathbb{T})} : \mathbb{T} \dashrightarrow \mathcal{C}(\mathbb{T})$ is a TSubseqTra-reflection arrow.*

Proof. We only provide a sketch. For item (1), let $\mathbb{T} = (Q, o, d, r, i, m)$ and $\mathcal{A}(\mathbb{T}) = (Acc(\mathbb{T}), o', d', r', i, m)$. To see that $id_{Acc(\mathbb{T})}$ satisfies (next), we have for all $q \in Acc(\mathbb{T})$ and all $a \in A$: $id_{Acc(\mathbb{T})}(d'(q)(a))$ is defined iff $a \in \mathrm{supp}(q)$ iff $d(id_{Acc(\mathbb{T})}(q))(a)$ is defined, and when both are defined they are equal, since d' is the restriction of d to $Acc(\mathbb{T})$. Condition (acc) holds since $F' := \mathrm{dom}(r') = F \cap Acc(\mathbb{T}) = id^{-1}_{Acc(\mathbb{T})}(F)$. The other conditions from Definition 3.4 are checked as easily. Item (2) can be proved along the same lines as Theorem 3.18. □

Since subsequential transducer morphisms respect behaviour (Proposition 3.9(2)) we have an easy corollary.

Corollary 3.22. *For any subsequential transducer $\mathbb{T}$, $\mathcal{C}(\mathcal{A}(\mathbb{T}))$ is equivalent to $\mathbb{T}$.*

Remark 3.23. In fact, it is also possible to show that for any subsequential transducer $\mathbb{T}$, $id_{Acc(\mathbb{T})} : \mathcal{A}(\mathbb{T}) \dashrightarrow \mathbb{T}$ is an ASubseqTra-coreflection arrow for $\mathbb{T}$. A coreflection arrow is the dual notion of a reflection arrow. Hence trimming a transducer can be seen as a composition of a coreflection with a reflection. However, we leave it to the interested reader to verify this claim, since it will not play any role in the rest of the paper.

3.4. Normalised subsequential structures

As we have seen in the previous section, considering coaccessible subsequential structures allows us to work with morphisms as total maps and unique witnessing functions. Still, in spite of this conceptual simplification, checking whether a morphism exists between two coaccessible structures, or whether two subsequential transducers are equivalent, is complicated by checking for the existence of a witnessing output shift function β. In this section, we will see that this problem is eliminated by considering normalised subsequential structures and transducers. Informally stated, a normalised subsequential transducer produces its output at maximal speed. Consequently, morphisms between normalised subsequential transducers can only have $\beta = \varepsilon$ as witnessing function.

The definition of normalised subsequential transducers goes back to Choffrut [9] who showed that any finite subsequential transducer can be transformed into an equivalent normalised one. Here we formulate Choffrut's results for coaccessible subsequential structures, and we show that normalised subsequential structures and transducers form reflective subcategories of CSubseq and CSubseqTra, respectively.

Definition 3.24 (*normalised states, structures and transducers*)**.** Let $\mathbb{S} = (Q, o, d, r)$ be a coaccessible subsequential structure, and $q \in Q$. We define a function $\hat{\beta}_{\mathbb{S}} : Q \to B^*$ by

$$\hat{\beta}_{\mathbb{S}}(q) = lcp(\{o(q)(w) \cdot r(d(q)(w)) \mid w \in \mathcal{L}(q)\}). \tag{4}$$

That is, $\hat{\beta}_{\mathbb{S}}(q)$ is the longest common prefix over all output words on final paths starting in q. A state $q \in Q$ *is normalised* if $\hat{\beta}_{\mathbb{S}}(q) = \varepsilon$. A subsequential structure $\mathbb{S}$ is *normalised* if all states in $\mathbb{S}$ are normalised, and a subsequential transducer $\mathbb{T}$ is *normalised* if its underlying subsequential structure is normalised.

If $\mathbb{T} = (\mathbb{S}, i, m)$, then will use the notation $\hat{\beta}_{\mathbb{T}} = \hat{\beta}_{\mathbb{S}}$, or we may leave out the subscript altogether if no confusion can arise. Let NSubseq be the full subcategory of CSubseq (and Subseq) consisting of normalised subsequential structures

and subsequential morphisms. Similarly, NSubseqTra is the full subcategory of CSubseqTra consisting of normalised subsequential transducers.

The meaning of $\hat{\beta}$ can be explained informally as follows. Suppose a subsequential transducer $\mathbb{T}$ is processing an input word $w = vu \in \mathrm{dom}([\![\mathbb{T}]\!])$, and after reading v, the output produced so far is $x \in B^*$ and the current state is q. Now $\hat{\beta}(q)$ gives us the longest word which will be output by $\mathbb{T}$ in the remainder of the computation, no matter what u is. This means that the output of $\hat{\beta}(q)$ is unnecessarily delayed while waiting for $\mathbb{T}$ to read the next input letter. Normalising a subsequential transducer consists in changing the output functions such that there is no delayed output anywhere.

Definition 3.25 (*normalisation of structures and transducers*)**.** Let $\mathbb{T} = (Q, o, d, r, i, m)$ be a coaccessible subsequential transducer. The *normalisation of* $\mathbb{T}$ is the subsequential transducer $\mathcal{N}(\mathbb{T}) = (Q, o', d, r', i, m')$ where for all $q \in Q$, and all $a \in A$:

$$m' = m \cdot \hat{\beta}(i), \quad o'(q)(a) = \overline{\hat{\beta}(q)} \cdot o(q)(a) \cdot \hat{\beta}(d(q)(a)), \quad r'(q) = \overline{\hat{\beta}(q)} \cdot r(q). \tag{5}$$

Similarly, for a subsequential structure $\mathbb{S} = (Q, o, d, r)$, the *normalisation of* $\mathbb{S}$ is $\mathcal{N}(\mathbb{S}) = (Q, o', d, r')$, where o' and r' are defined as in (5).

Lemma 3.26. *If $\mathbb{S}$ is a coaccessible subsequential structure then $\mathcal{N}(\mathbb{S})$ is a normalised subsequential structure. Consequently, for all coaccessible subsequential transducers $\mathbb{T}$, $\mathcal{N}(\mathbb{T})$ is a normalised subsequential transducer.*

Proof. First note that in Definition 3.25, $o'(q)(a)$ and $r'(q)$ take values in B^* for all states q and $a \in A$, since $\hat{\beta}(q)$ is a prefix of $o(q)(a) \cdot \hat{\beta}(d(q)(a))$ and of $r(q)$. So $\mathcal{N}(\mathbb{S})$ is a subsequential structure. The fact that $\mathcal{N}(\mathbb{S})$ is normalised is clear from the definition of $\hat{\beta}$. □

Example 3.27. The reader can verify that in Example 3.3, $\mathbb{T}_2$ is the normalisation of $\mathbb{T}_1$, and the β given in Example 3.6 is equal to $\hat{\beta}_{\mathbb{T}_1}$.

Before we characterise subsequential morphisms between normalised structures, we note that from Proposition 3.9 it follows that if $(\alpha, \beta) : \mathbb{S}_1 \to \mathbb{S}_2$ in CSubseq, then for all states q in $\mathbb{S}_1$:

$$\hat{\beta}_1(q) = \beta(q) \cdot \hat{\beta}_2(\alpha(q)), \tag{6}$$

where $\hat{\beta}_j = \hat{\beta}_{\mathbb{S}_j}, j = 1, 2$. From (6) it follows immediately that:

Lemma 3.28. *Let $(\alpha, \beta) : \mathbb{S}_1 \to \mathbb{S}_2$ in* CSubseq.

1. *If $\mathbb{S}_2$ is normalised, then $\beta = \hat{\beta}_1$.*
2. *If $\mathbb{S}_1$ and $\mathbb{S}_2$ are normalised, then $\beta = \varepsilon$.*

We can now prove that subsequential morphisms between normalised structures are very much like morphisms between deterministic automata or Mealy machines.

Proposition 3.29. *Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be normalised subsequential structures. A function $\alpha : Q_1 \to Q_2$ is a subsequential morphism if and only if $(\alpha, \varepsilon) : \mathbb{S}_1 \to \mathbb{S}_2$, i.e., for all $q \in Q_1$:*

(supp)	$\mathrm{supp}(q) = \mathrm{supp}(\alpha(q))$,
$(\mathrm{next})_{\mathsf{C}}$	$\forall a \in \mathrm{supp}(q) : \alpha(d_1(q)(a)) = d_2(\alpha(q))(a)$,
$(\mathrm{out})_{\mathsf{N}}$	$\forall a \in \mathrm{supp}(q) : o_1(q)(a) = o_2(\alpha(q))(a)$,
(acc)	$\alpha^{-1}(F_2) = F_1$,
$(\text{term-out})_{\mathsf{N}}$	*if* $q \in F_1$ *then* $r_1(q) = r_2(\alpha(q))$.

Let $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_1)$ and $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ be normalised subsequential transducers. A subsequential (transducer) morphism from $\mathbb{T}_1$ to $\mathbb{T}_2$ is a subsequential morphism $\alpha : \mathbb{S}_1 \to \mathbb{S}_2$ satisfying:

(init)	$\alpha(i_1) = i_2$, *and*
$(\varepsilon\text{-in})_{\mathsf{N}}$	*if* $i_1 \in \mathrm{dom}(\alpha)$ *then* $m_1 = m_2$.

Proof. First consider the characterisation of subsequential morphisms between normalised structures. If $\alpha : \mathbb{S}_1 \to \mathbb{S}_2$, then by Lemma 3.28(2) the (unique) witnessing function is $\beta = \varepsilon$. The other direction is clear. The characterisation of morphisms between normalised transducers follows easily from the result for structures. □

An easy consequence of Proposition 3.29 is that subsequential morphisms between normalised structures preserve state behaviour.

Corollary 3.30. *Let $\alpha : \mathbb{S}_1 \to \mathbb{S}_2$ be a subsequential morphism in* NSubseq. *For all states q in $\mathbb{S}_1$: $[\![q]\!]_1 = [\![\alpha(q)]\!]_2$.*

Proof. From Proposition 3.9 we have for all q in $\mathbb{S}_1$ that $[\![q]\!]_1 = \beta(q) \cdot [\![\alpha(q)]\!]_2$. From Proposition 3.29 it follows that $\beta(q) = \varepsilon$, and hence $[\![q]\!]_1 = [\![\alpha(q)]\!]_2$. □

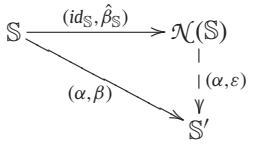
We now show that normalisation is a reflector. This result takes the argument from the previous section for coaccessible structures and transducers a step further, so that we now can say that the right way of thinking about subsequential structures and transducers is in terms of their normalisation.

Theorem 3.31. *Let $\mathbb{S}$ be a coaccessible subsequential structure and $\mathbb{T} = (\mathbb{S}, i, m)$ a coaccessible subsequential transducer. We have:*

1. *$id_{\mathbb{S}} : \mathbb{S} \to \mathcal{N}(\mathbb{S})$ is an* NSubseq*-reflection arrow for $\mathbb{S}$.*
2. *$id_{\mathbb{T}} : \mathbb{T} \to \mathcal{N}(\mathbb{T})$ is an* NSubseqTra*-reflection arrow for $\mathbb{T}$.*

It follows that NSubseq *is a reflective subcategory of* CSubseq, *and* NSubseqTra *is a reflective subcategory of* CSubseqTra. *Moreover, the map $\mathcal{N}$ is a functor $\mathcal{N}$:* CSubseq $\to$ NSubseq *and $\mathcal{N}$:* CSubseqTra $\to$ NSubseqTra *by defining $\mathcal{N}(\alpha) = \alpha$ for a subsequential morphism α in* CSubseq *or* CSubseqTra .

Proof. Item (1): Let $\mathbb{S}$ be an object in CSubseq. It is straightforward to check that $(id_{\mathbb{S}}, \hat{\beta}_{\mathbb{S}}) : \mathbb{S} \to \mathcal{N}(\mathbb{S})$ by using the characterisation of morphisms in CSubseq (Proposition 3.16) and the definition of $\mathcal{N}(\mathbb{S})$ (Definition 3.25). Now it is also easy to see that for any normalised $\mathbb{S}' \in$ NSubseq, if $(\alpha, \beta) : \mathbb{S} \to \mathbb{S}'$ in CSubseq then $\alpha' = \alpha$ is the unique NSubseq-morphism such that $\alpha' : \mathcal{N}(\mathbb{S}) \to \mathbb{S}'$ and $\alpha' \circ id_{\mathbb{S}} = \alpha$. From Lemma 3.28(2) it follows that the witnessing function β' for $\alpha : \mathcal{N}(\mathbb{S}) \to \mathbb{S}'$ is just $\beta' = \varepsilon$, and that the diagram below commutes.



Item (2): Let $\mathbb{T} = (\mathbb{S}, i, m,)$ be in CSubseqTra. Hence in particular, $i \in Coacc(\mathbb{T})$. It is easy to check that $(id_{\mathbb{T}}, \hat{\beta}_{\mathbb{T}}) : \mathbb{T} \to \mathcal{N}(\mathbb{T})$ in CSubseqTra. Now suppose $(\alpha, \beta) : \mathbb{T} \to \mathbb{T}'$ for some $\mathbb{T}' = (\mathbb{S}', i', m')$ in NSubseqTra. It follows that, $(\alpha, \beta) : \mathbb{S} \to \mathbb{S}'$, hence by item (1) $\alpha : \mathcal{N}(\mathbb{S}) \to \mathbb{S}'$ is the unique subsequential morphism such that $\alpha = \alpha \circ id_{\mathbb{S}}$. To see that $\alpha : \mathcal{N}(\mathbb{T}) \to \mathbb{T}'$ in NSubseqTra, note that $(\alpha, \beta) : \mathbb{T} \to \mathbb{T}'$ implies that $m' = m \cdot \beta(i)$, and using Lemma 3.28(1) we get $m' = m \cdot \hat{\beta}_{\mathbb{S}}(i)$. Hence $\mathcal{N}(\mathbb{T})$ and $\mathbb{T}'$ have the same prefix. The uniqueness of α follows from the uniqueness of α on the underlying structures. □

Corollary 3.32. *For all $\mathbb{T}$ in* CSubseqTra, *we have: $[\![\mathbb{T}]\!] = [\![\mathcal{N}(\mathbb{T})]\!]$.*

Proof. Follows from Theorem 3.31(2), and the fact that subsequential morphisms preserve transducer behaviour. □

Choffrut surveys in [10] a number of different algorithms for computing $\hat{\beta}$. One of these algorithms is by Béal and Carton [4] who report that for a normalised $\mathbb{S}$, $\hat{\beta}$ can be computed in time $O((\|\hat{\beta}\| + 1)M)$, where $\|\hat{\beta}\|$ is the maximal length of $\hat{\beta}(q)$ over all states q in $\mathbb{S}$, and M is the number of transitions in $\mathbb{S}$.

3.5. Minimal subsequential transducers

A subsequential structure $\mathbb{S}$ is called *minimal*, if $\mathbb{S}$ is normalised and no two states in $\mathbb{S}$ are equivalent. A subsequential transducer is minimal if its underlying structure is minimal. Choffrut remarked in [10, p. 131 and 139] that minimisation of normalised subsequential transducers can be carried out by generalising existing techniques for minimising determin-

istic automata [20,24], however, the details were not given.[4] We describe the minimisation of normalised structures and transducers in Section 4.3.

Choffrut showed also in [10] that for any function $f : A^* \dashrightarrow B^*$, there exists a minimal (but possibly infinite) subsequential transducer $\mathbb{T}_f$ with behaviour f such that for all trimmed subsequential transducers $\mathbb{T}$ which also realise f, there is a unique subsequential morphism $\alpha : \mathbb{T} \to \mathbb{T}_f$. This result strongly suggests the existence of a final subsequential structure, and in the next section we will prove that indeed the existence and properties of $\mathbb{T}_f$ follow from finality, (Corollary 4.13).

As expected, minimal subsequential structures form a reflective subcategory of normalised subsequential structures.

Proposition 3.33. *The full subcategory of minimal subsequential structures (transducers) is reflective in* NSubseq *(*NSubseqTra*).*

Proof. The result for structures follows from Proposition 4.4 of the next section, which shows that NSubseq is a full subcategory of Coalg($\mathcal{S}$), together with a more general result from [16], where it is proved that for any functor $\mathcal{F}$, minimal $\mathcal{F}$-coalgebras are reflective in Coalg($\mathcal{F}$). To get a reflector for NSubseq, we just have to restrict the reflector for Coalg($\mathcal{S}$) to the full subcategory NSubseq. The argument for the transducer case is almost identical. □

4. Coalgebraisation via normalisation

One of our aims is to find out whether subsequential structures can be seen as coalgebras. A fundamental property of a category of coalgebras is that it comes with abstract definitions of morphism and state behaviour, which capture the general idea that coalgebra morphisms are behaviour preserving maps. In order to claim that a class of subsequential structures is coalgebraic, we would want the notions of subsequential morphism and state behaviour (defined in Section 3.1) to coincide with the coalgebraic ones. However, we already suspect that, in general, subsequential structures and morphisms are not coalgebraic, since subsequential morphisms do not preserve state behaviour, unless we find ourselves in the subcategory of normalised structures.

In this section we will first demonstrate that indeed the category NSubseq can be regarded as a category of coalgebras, whereas this does not hold for Subseq and CSubseq. This result is the basis for our slogan that *normalisation is coalgebraisation*. In Section 4.2 we prove that NSubseq has a final object. Since NSubseq is reflective in Subseq, it follows that the final NSubseq-object is also a final Subseq-object. We also show that the existence and properties of minimal subsequential transducers (cf. Section 3.5) follow from the existence of this final object. In Section 4.3 we describe how to minimise normalised structures by adapting the standard minimisation algorithm for deterministic finite automata.

4.1. Coalgebraic modelling

First recall that a partial function $f : X \dashrightarrow Y$ can be seen as a total function $f : X \to \{\star\} \cup Y$ where $\star$ is the undefined value by taking $f(x) = \star$, if $x \notin \mathrm{dom}(f)$. We will use the notation $\mathbf{1} = \{\star\}$ and write $\mathbf{1} + Y$ instead of $\{\star\} \cup Y$.

Let $\mathbb{S} = (Q, o, d, r)$ be a subsequential structure. We combine o and d into a transition structure $t : Q \to (A \to (\mathbf{1} + B^* \times Q))$ by defining for all $q \in Q$: $t(q)(a) = \langle o(q)(a), d(q)(a)\rangle$ if $a \in \mathrm{supp}(q)$; otherwise $t(q)(a) = \star$. Similarly, we view $r : Q \dashrightarrow B^*$ as a total function $r : Q \to \mathbf{1} + B^*$. It is then easy to see that $\mathbb{S}$ can be fully described by a single map:

$$\begin{array}{rcl} \langle t, r\rangle & : & Q \to (\mathbf{1} + B^* \times Q)^A \times (\mathbf{1} + B^*) \\ & & q \mapsto \qquad\qquad \langle\ t(q)\ ,\ r(q)\ \rangle. \end{array} \tag{7}$$

This map has the type of a coalgebra for the functor $\mathcal{S} : \mathsf{Set} \to \mathsf{Set}$ defined by:

$$\begin{array}{rcl} \mathcal{S}(X) & = & (\mathbf{1} + B^* \times X)^A \times (\mathbf{1} + B^*), \\ \mathcal{S}(f : X \to Y) & = & (\mathbf{1} + id_{B^*} \times f)^{id_A} \times (\mathbf{1} + id_{B^*}). \end{array}$$

Clearly, every map $\langle t, r\rangle$ of the type given in (7) can also be seen as a subsequential structure, and from now on we will make no distinction between the two. Instantiating the definition of $\mathcal{S}$-coalgebra morphism yields the following. Let $\mathbb{X}_1 = (X_1, \langle t_1, r_1\rangle)$ and $\mathbb{X}_2 = (X_2, \langle t_2, r_2\rangle)$ be $\mathcal{S}$-coalgebras. A function $\alpha : X_1 \to X_2$ is an $\mathcal{S}$-coalgebra morphism from $\mathbb{X}_1$ to $\mathbb{X}_2$ if for all $x \in X_1$: $\mathcal{S}(\alpha)(\langle t_1(x), r_1(x)\rangle) = \langle t_2(\alpha(x)), r_2(\alpha(x))\rangle$, which is equivalent with:

[4] Choffrut states in [10, p. 139] that a normalised subsequential transducer can be minimised by minimising the underlying automaton. This is however not true (and we assume it was just a misformulation by Choffrut), since it is clearly possible that two states are equivalent with respect to the underlying deterministic automaton while not having the same behaviour.

(T1) $\forall a \in A : t_1(x)(a) = \star \iff t_2(\alpha(x))(a) = \star$,

(T2) $\forall a \in A : t_1(x)(a) \neq \star \Rightarrow$
$o_1(x)(a) = o_2(\alpha(x))(a)$ and $\alpha(d_1(x)(a)) = d_2(\alpha(x))(a)$,

(R1) $r_1(x) = \star \iff r_2(\alpha(x)) = \star$,

(R2) $r_1(x) \neq \star \implies r_1(x) = r_2(\alpha(x))$.

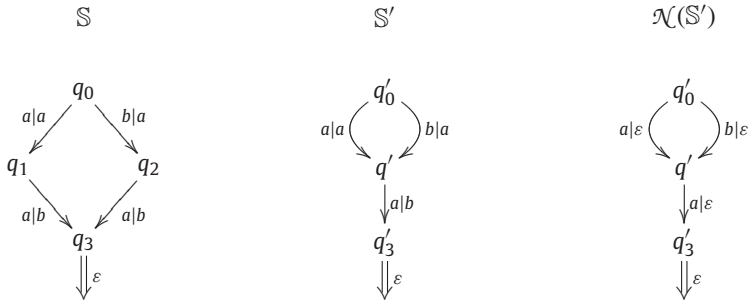
The notion of $\mathcal{S}$-coalgebra morphism applies to arbitrary subsequential structures. The following lemma tells us when a subsequential morphism is also an $\mathcal{S}$-coalgebra morphism.

Lemma 4.1. *Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be two subsequential structures ($\mathcal{S}$-coalgebras), and $\alpha : Q_1 \to Q_2$ a total function. We have: α is an $\mathcal{S}$-coalgebra morphism from $\mathbb{S}_1$ to $\mathbb{S}_2$ if and only if $\alpha : \mathbb{S}_1 \to \mathbb{S}_2$ is a subsequential morphism with witness $\beta = \varepsilon$.*

Proof. Using the assumption that α is total, it is easily verified that the conditions (T1), (T2), (R1) and (R2) are equivalent with the conditions of Definition 3.4 for witnessing function $\beta = \varepsilon$. □

The following example shows that an $\mathcal{S}$-coalgebra morphism can exist between non-normalised subsequential structures.

Example 4.2. Consider the non-normalised subsequential structures $\mathbb{S}$ and $\mathbb{S}'$ depicted below together with the normalisation of $\mathbb{S}'$.



The map α defined by $\alpha(q_0) = q'_0$, $\alpha(q_1) = \alpha(q_2) = q'$, and $\alpha(q_3) = q'_3$, is an $\mathcal{S}$-coalgebra morphism between the coalgebras induced by $\mathbb{S}$ and $\mathbb{S}'$. Hence by Lemma 4.1, α is also a subsequential morphism with witnessing function $\beta = \varepsilon$, as can easily be checked directly. On the other hand, the identity map on the states is a subsequential morphism from $\mathbb{S}'$ to $\mathcal{N}(\mathbb{S}')$ (cf. Theorem 3.31), but not an $\mathcal{S}$-coalgebra morphism.

One immediate consequence of Lemma 4.1 is that $\mathcal{S}$-bisimilarity always implies state equivalence.

Proposition 4.3. *Let $\mathbb{S}$ be a subsequential structure containing states s_1 and s_2. If $s_1 \sim_{\mathcal{S}} s_2$ then $[\![s_1]\!] = [\![s_2]\!]$.*

Proof. Let $\mathbb{S}$, s_1 and s_2 be as stated. Assume $(s_1, s_2) \in Z$ for some $\mathcal{S}$-bisimulation (Z, ζ) on $\mathbb{S}$. It follows that the projections $\pi_1, \pi_2 : Z \to S$ are $\mathcal{S}$-coalgebra morphisms. By Lemma 4.1, the projections are subsequential morphisms with witness ε, and from Proposition 3.9, we get $[\![s_1]\!]_{\mathbb{S}} = [\![(s_1, s_2)]\!]_{(Z,\zeta)} = [\![s_2]\!]_{\mathbb{S}}$. □

In the following proposition we make precise what we mean by saying that normalised subsequential structures can be properly regarded as coalgebras.

Proposition 4.4. $\mathsf{NSubseq}$ *is a full subcategory of* $\mathsf{Coalg}(\mathcal{S})$.

Proof. Follows from the characterisation of $\mathsf{NSubseq}$-morphisms in Proposition 3.29 and Lemma 4.1. □

4.2. The final subsequential structure

In a category of coalgebras, a final object can be thought of as the coalgebra of behaviours. In the case of normalised subsequential structures, state behaviours are functions $f : A^* \dashrightarrow B^*$ with the property that all words in the range of f have only a trivial common prefix, i.e., $lcp(f(A^*)) = \varepsilon$. We call an f with this property *normalised*. We can define a normalised subsequential structure on the set of normalised functions using the notions of maximal output and derivative.

These operations also form the basis of Choffrut's construction in [10] of the minimal transducer realising a function f.[5] Informally, given a function $f\colon A^* \dashrightarrow B^*$ and a word $w \in A^*$, the maximal output of f on w is the longest common prefix over all output words generated by f on inputs that start with w. The derivative of f with respect to w is the function which maps an input word u to the output word obtained by removing the maximal output on w from $f(wu)$, given that $f(wu)$ is defined. We point out that other types of derivatives of formal languages, streams and stream functions have been shown to yield final coalgebra structures (cf. [35,38,39]). We now give the formal definition.

Definition 4.5. Let $f\colon A^* \dashrightarrow B^*$ and $w \in A^*$. The *maximal output of f on input w* is defined as

$$f[w] := lcp(f(wA^*)) = lcp(\{f(wu) \mid wu \in \mathrm{dom}(f)\}).$$

The *(word function) derivative of f with respect to w* is the partial function $f\cdot w\colon A^* \dashrightarrow B^*$ defined for all $u \in A^*$ by

$$(f\cdot w)(u) = \begin{cases} \overline{f[w]}\,.\,f(wu) & \text{if } wu \in \mathrm{dom}(f) \\ \star & \text{otherwise} \end{cases}.$$

Note that $lcp(f(A^*)) = f[\varepsilon]$, i.e., f is normalised iff $f[\varepsilon] = \varepsilon$. We observe that derivatives are normalised.

Lemma 4.6. *For all $f\colon A^* \dashrightarrow B^*$ and all $w \in A^*$, $(f\cdot w)[\varepsilon] = \varepsilon$.*

Proof. We have:

$$\begin{aligned} (f\cdot w)[\varepsilon] &= lcp(\{(f\cdot w)(u) \mid u \in \mathrm{dom}(f\cdot w)\}) \\ &= lcp(\{\overline{f[w]}\,.\,f(wu) \mid u \in \mathrm{dom}(f\cdot w)\}) \\ &= \overline{f[w]}\,.\,lcp(\{f(wu) \mid wu \in \mathrm{dom}(f)\}) \\ &= \overline{f[w]}\,.\,f[w] = \varepsilon. \quad \square \end{aligned}$$

Definition 4.7. We define $\Omega := (Q_\Omega, O, D, R)$ where

$$Q_\Omega := \{f\colon A^* \dashrightarrow B^* \mid f[\varepsilon] = \varepsilon\},$$

and for all $f \in Q_\Omega$ and $a \in A$:

$$O(f)(a) = f[a], \quad D(f)(a) = f\cdot a, \quad R(f) = f(\varepsilon).$$

We first show that Ω is an object in $\mathsf{NSubseq}$.

Lemma 4.8. *$\Omega = (Q_\Omega, O, D, R)$ is a normalised subsequential structure.*

Proof. First note that D, and hence Ω, is well-defined due to Lemma 4.6. To prove that Ω is normalised, we show by induction on $|w|$ that for all $f \in Q_\Omega$ and $w \in A^*$, $[\![f]\!]_\Omega(w) = f(w)$. The base case follows from the definition of R. Now let $a \in A$, $w \in A^*$ and $f \in Q_\Omega$. We have:

$$[\![f]\!]_\Omega(aw) = f[a]\,.\,[\![f\cdot a]\!]_\Omega(w) =_{(\text{by IH})} f[a]\,.\,(f\cdot a)(w) = f[a]\,.\,\overline{f[a]}\,.\,f(aw) = f(aw).$$

It follows that f is normalised as a state in Ω, since then $\hat{\beta}(f) = [\![f]\!]_\Omega[\varepsilon] = f[\varepsilon]$. $\square$

We now show that Ω is final in $\mathsf{NSubseq}$ with the behaviour map $[\![\_]\!]$ as the unique subsequential morphism into Ω.

Theorem 4.9. *The normalised subsequential structure Ω is a final object in the category* $\mathsf{NSubseq}$.

Proof. Let $\mathbb{S} = (Q, o, d, r)$ be a normalised subsequential structure, and let $[\![\_]\!] = [\![\_]\!]_{\mathbb{S}}\colon Q \to Q_\Omega$ be the state behaviour map. We will show that $[\![\_]\!]$ is a subsequential morphism, i.e., an $\mathcal{S}$-coalgebra morphism. First of all, since all states in $\mathbb{S}$ are coaccessible, $[\![\_]\!]$ is a total function. We now check that $[\![\_]\!]$ satisfies the conditions from Proposition 3.29. Let $q \in Q$, and $a \in A$. We have: $a \in \mathrm{supp}(q)$ iff $aw \in \mathrm{dom}([\![q]\!])$ for some $w \in A^*$ iff $\mathrm{dom}([\![q]\!]\cdot a) \neq \emptyset$ iff $a \in \mathrm{supp}([\![q]\!])$.

[5] Choffrut's definition is based on the congruence classes of the syntactic congruence of f, but it can easily be reformulated in terms of derivatives of word functions (also called the residual).

To see that $O([\![q]\!])(a) = [\![q]\!][a] = o(q)(a)$, we note that for all $w \in A^*$ such that $aw \in \mathrm{dom}([\![q]\!])$, $[\![q]\!](aw) = o(q)(a) \cdot [\![d(q)(a)]\!](w)$, and hence

$$[\![q]\!][a] = lcp([\![q]\!](aA^*)) = o(q)(a) \cdot lcp([\![d(q)(a)]\!](A^*)) = o(q)(a) \cdot \varepsilon = o(q)(a),$$

since $d(q)(a)$ is normalised. In order to show that $[\![d(q)(a)]\!] = [\![q]\!]\cdot a$ for all $a \in \mathrm{supp}(q)$, let $w \in A^*$. We then have

$$\begin{aligned}
[\![d(q)(a)]\!](w) &= o(d(q)(a))(w) \cdot r(d(d(q)(a))(w)) \\
&= \overline{o(q)(a)} \cdot o(q)(aw) \cdot r(d(q)(aw)) \\
&= \overline{[\![q]\!][a]} \cdot [\![q]\!](aw) \\
&= ([\![q]\!]\cdot a)(w).
\end{aligned}$$

Finally, we have $q \in \mathrm{dom}(r)$ iff $\varepsilon \in \mathrm{dom}([\![q]\!])$ iff $[\![q]\!] \in \mathrm{dom}(R)$, and $R([\![q]\!]) = [\![q]\!](\varepsilon) = r(q)$. We leave it to the reader to verify that $[\![\_]\!] : (Q, o, d, r) \to (Q_\Omega, O, D, R)$ is unique. □

Since NSubseq is reflective in Subseq, Ω is also final in Subseq.

Corollary 4.10. *The normalised subsequential structure Ω is a final object in* Subseq.

Proof. This is an immediate consequence of Theorem 4.9 and the fact that NSubseq is reflective in Subseq, which follows from Theorems 3.31 and 3.18 and the fact that reflectors compose. For $\mathbb{S} \in$ Subseq, the final Subseq-morphism $h_\mathbb{S} : \mathbb{S} \to \Omega$ is obtained by composing the reflection arrows with the behaviour map in NSubseq:

$$h_\mathbb{S} = [\![\_]\!]_{\mathcal{N}(C(\mathbb{S}))} \circ id_{C(\mathbb{S})} \circ id_{Coacc(\mathbb{S})}.$$

Concretely, one can show that $h_\mathbb{S} : \mathbb{S} \to \Omega$ is the partial map defined for all $q \in Coacc(\mathbb{S})$ by $h_\mathbb{S}(q) = [\![q]\!]_\mathbb{S}\cdot\varepsilon = \overline{\hat{\beta}_\mathbb{S}(q)} \cdot [\![q]\!]_\mathbb{S}$. As witnessing function take any β which agrees with $\hat{\beta}_{C(\mathbb{S})}$ on $Coacc(\mathbb{S})$. □

Using the fact that behaviour is a morphism, we can show that in normalised structures, state equivalence coincide with $\mathcal{S}$-bisimilarity.

Corollary 4.11. *Let $\mathbb{S}$ be a normalised subsequential structure. For all states q_1, q_2 in $\mathbb{S}$, we have: $[\![q_1]\!]_\mathbb{S} = [\![q_2]\!]_\mathbb{S}$ if and only if $q_1 \sim_\mathcal{S} q_2$.*

Proof. First assume that $[\![q_1]\!]_\mathbb{S} = [\![q_2]\!]_\mathbb{S}$. Since $\mathbb{S}$ is normalised, $[\![\_]\!]_\mathbb{S} : \mathbb{S} \to \Omega$ is also an $\mathcal{S}$-coalgebra morphism, and it follows from Proposition 2.3(2) that $q_1 \sim_\mathcal{S} q_2$. The other direction holds by Proposition 4.3. □

Note that Ω is not final in $\mathrm{Coalg}(\mathcal{S})$, since for an arbitrary $\mathbb{S}$ in $\mathrm{Coalg}(\mathcal{S})$, the final Subseq-morphism will not be an $\mathcal{S}$-coalgebra morphism (cf. Lemma 3.28(1)). However, since $\mathcal{S}$ is polynomial, we know that a final $\mathcal{S}$-coalgebra Γ exists (cf. Proposition 2.3(1)). Corollary 4.11 tells us that Ω is minimal not only with respect to state equivalence, but also with respect to $\mathcal{S}$-bisimilarity. Consequently, Ω can be considered a subcoalgebra of Γ.

Theorem 4.12. *The final subsequential structure Ω is isomorphic to a subcoalgebra of the final $\mathcal{S}$-coalgebra Γ.*

Proof. Let ϕ_Ω be the final $\mathrm{Coalg}(\mathcal{S})$-morphism from Ω to Γ. We will show that ϕ_Ω is injective. It then follows that Ω is isomorphic to the subcoalgebra $\phi_\Omega(\Omega)$ of Γ. So assume $\phi_\Omega(f) = \phi_\Omega(g)$ for $f, g \in Q_\Omega$. It follows from Proposition 2.3(2) that $f \sim_\mathcal{S} g$, hence by Proposition 4.3 we have $[\![f]\!]_\Omega = [\![g]\!]_\Omega$, and by finality of Ω, we conclude that $f = g$. □

We now show that the existence and properties of a minimal transducer $\mathbb{T}_f$ realising a function $f : A^* \dashrightarrow B^*$ given in [10] are a consequence of Theorem 4.9. Recall that $\langle f\rangle_\Omega$ denotes the subcoalgebra generated by f in Ω.

Corollary 4.13. *Let $f : A^* \dashrightarrow B^*$ be any partial function.*

1. *$\mathbb{T}_f = (\langle f\cdot\varepsilon\rangle_\Omega, f\cdot\varepsilon, f[\varepsilon])$ is a minimal subsequential transducer with behaviour f.*
2. *If $\mathbb{T}$ is an accessible subsequential transducer with $[\![\mathbb{T}]\!] = f$, then there is a unique subsequential transducer morphism from $\mathbb{T}$ to $\mathbb{T}_f$.*

Proof. Item (1): The subcoalgebra $\langle f\cdot\varepsilon\rangle_\Omega$, and hence $\mathbb{T}_f$, is minimal since Ω is minimal. To see that $[\![\mathbb{T}_f]\!] = f$, we have for all $w \in A^*$, $w \in \mathrm{dom}(f)$ iff $w \in \mathrm{dom}(f\cdot\varepsilon) = \mathrm{dom}(\mathbb{T}_f)$, and for $w \in \mathrm{dom}(f)$ we have:

$$[\![\mathbb{T}_f]\!](w) = f[\varepsilon] \cdot [\![f\cdot\varepsilon]\!]_\Omega(w) = f[\varepsilon] \cdot (f\cdot\varepsilon)(w) = f[\varepsilon] \cdot \overline{f[\varepsilon]} \cdot f(w) = f(w).$$

Item (2): Let $\mathbb{T} = (\mathbb{S}, i, m)$ be accessible. The final Subseq-map $h_{\mathbb{S}}$ is a subsequential morphism from $\mathbb{S}$ to $\langle f \cdot \varepsilon \rangle_\Omega$ with a witnessing function β such that $\beta \restriction_{Coacc(\mathbb{S})} = \hat{\beta}_{C(\mathbb{S})}$ (cf. Corollary 4.10). In the case the initial state of $\mathbb{T}$ is not coaccessible, $\mathbb{T}_f$ is the empty transducer, and $h_{\mathbb{S}}$ is the empty map. If $i \in Coacc(\mathbb{T})$, it follows that $h_{\mathbb{S}}$ and β also satisfy (ε-in) since $f[\varepsilon] = [\![\mathbb{T}]\!][\varepsilon] = m \cdot \hat{\beta}_{\mathbb{S}}(i)$, and (init) since:

$$h_{\mathbb{S}}(i) = \overline{\hat{\beta}_{\mathbb{S}}(i)} \cdot [\![i]\!]_{\mathbb{S}} = \overline{\hat{\beta}_{\mathbb{S}}(i)} \cdot \overline{m} \cdot f = \overline{m \cdot \hat{\beta}_{\mathbb{S}}(i)} \cdot f = \overline{f[\varepsilon]} \cdot f = f \cdot \varepsilon.$$

Uniqueness follows from uniqueness of $h_{\mathbb{S}}$. □

4.3. Minimisation algorithm for normalised structures

Due to Propositions 2.3(3) and Corollary 4.11, normalised subsequential structures can be minimised by quotienting with $\mathcal{S}$-bisimilarity. We now describe how we can compute $\mathcal{S}$-bisimilarity on an $\mathcal{S}$-coalgebra (and hence on a normalised subsequential structure) by adapting the existing method for computing state equivalence on deterministic finite automata (DFA) (see e.g. ([20,24])).

First of all, working out the details of the definition of $\mathcal{S}$-bisimulation yields the following. Given an $\mathcal{S}$-coalgebra $\mathbb{S} = (Q, o, d, r)$, a relation $R \subseteq Q \times Q$ is an $\mathcal{S}$-bisimulation on $\mathbb{S}$, if for any two states $q, s \in Q$, $\langle q, s \rangle \in R$ implies

(s0) $\mathrm{supp}(q) = \mathrm{supp}(s)$;
(s1) for all $a \in \mathrm{supp}(q) : o(q)(a) = o(s)(a)$;
(s2) for all $a \in \mathrm{supp}(q) : \langle d(q)(a), d(s)(a) \rangle \in R$; and
(s3) $r(q) = r(s)$.

In order to make the connection with the algorithm for DFAs clear, we also give the definition of bisimulation which applies to DFAs. Recall (cf. Example 2.2) that a deterministic automaton (Q, d, F) is a coalgebra $\langle o, d \rangle : Q \to \mathbf{2} \times Q^A$ for the functor $\mathcal{A}ut(X) = \mathbf{2} \times Q^A$. Let $\mathbb{A} = (Q, d, F)$ be a finite $\mathcal{A}ut$-coalgebra. A relation $R \subseteq Q \times Q$ is an $\mathcal{A}ut$-bisimulation on $\mathbb{A}$, if for any two states $q, s \in Q$, $\langle q, s \rangle \in R$ implies that:

(a1) $o(q) = o(s)$ (i.e. $q \in F$ iff $s \in F$); and
(a2) for all $a \in A : \langle d(q)(a), d(s)(a) \rangle \in R$.

We now briefly sketch the algorithm for computing $\mathcal{A}ut$-bisimilarity in a finite deterministic automaton $\mathbb{A} = (Q, d, F)$. The computation starts with the partition $P_0 = \{F, Q \setminus F\}$ of Q. P_0 is the largest equivalence relation such that condition (a1) holds for all P_0-related states. In the main loop, P_0 is iteratively refined into an equivalence relation which also satisfies condition (a2). This is done by inspecting the current partition $P_k = \{Q_1, \ldots, Q_n\}$ of Q for the existence of some $Q_i, Q_j \in P_k$ and $a \in A$ such that there are $q, s \in Q_i$ for which $d(q)(a) \in Q_j$ and $d(s)(a) \notin Q_j$. In that case, Q_i is split (by (Q_j, a)) into the two sets $Q_i' = \{q \in Q_i \mid d(q)(a) \in Q_j\}$ and $Q_i'' = \{q \in Q_i \mid d(q)(a) \notin Q_j\}$, in other words, P_k is refined into $P_{k+1} = (P_k \setminus \{Q_i\}) \cup \{Q_i', Q_i''\}$. This refinement process continues until no more splits can be made. When this happens, the partition stores the $\mathcal{A}ut$-bisimilarity classes on $\mathbb{A}$. By using extra data structures it is possible to choose the *splitters* (Q_j, a) wisely, and reduce the number of actual splits that must be carried out, resulting in an algorithm which runs in time $O(|A|\, n \log(n))$ where n is the number of states, and $|A|$ is the size of the input alphabet A (cf. [23], see also [14,19]).

The adaptation of the DFA-algorithm to $\mathcal{S}$-coalgebras consists of changing the initial partition. The refinement part of the algorithm stays the same. We take as initial partition the (classes of) the largest equivalence relation P_0^s on Q such that all pairs related by P_0^s satisfy (s0), (s1) and (s3). Running the refinement algorithm starting from this initial partition will result in the largest equivalence relation which satisfies (s0)–(s3), i.e., the bisimilarity relation on $\mathbb{S}$. This can be proved by essentially the same argument used for the correctness of the algorithm for DFAs, see e.g. [23, Proof of Proposition 5]. In the next lemma we describe how to compute P_0^s.

Lemma 4.14. *Given a finite $\mathcal{S}$-coalgebra $\mathbb{S} = (Q, o, d, r)$, we can compute the largest equivalence relation P_0^s on Q which satisfies (s0), (s1) and (s3) in time $O((|A|\|o\| + \|r\|)|Q| \log(|Q|))$, where $\|o\| := \max\{|o(q)(a)| \mid q \in Q, a \in A\}$ and $\|r\| := \max\{|r(q)| \mid q \in Q\}$.*

Proof. We want to group together states which have the same output function and the same terminal output. This can be done efficiently by a variation on a sorting algorithm which can be implemented using a balanced binary search tree (cf. [22]). The nodes of a binary search tree T are pairs (c, l_c) where c is a key and l_c is a value, and it is required that a linear order $<^T$ exists on the set of all keys. Since $\mathbb{S}$ is finite, we can assume $A = \{a_1, a_2, \ldots, a_k\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ by enumerating all input and output letters that occur in $\mathbb{S}$. In our case, a key c is a data record which stores the output and terminal output functions for some state q. We define $c(q) := \langle o(q)(a_1), \ldots, o(q)(a_k), r(q) \rangle$. The value l_c associated with a key c will be a list of states q such that $c(q) = c$. We will use a variation on insertion which does the following. Inserting $(c(q), q)$ into a tree which contains a node $(c', l_{c'})$ with $c' = c(q)$ will result in adding q to l_c, and if $c(q)$ does not occur in

the tree, then a new node $(c(q), \{q\})$ is added. By inserting $(c(q), q)$ for all states q into an initially empty tree. we can obtain the P_0^S-equivalence classes by traversing the resulting tree and retrieving the node values l_c.

It remains to define a linear ordering on the key values, which are elements of $(\mathbf{1} + B^*)^{k+1}$. We define a linear order $\prec$ on B by $b_1 \prec b_2 \prec \ldots \prec b_n$ and extend this ordering to the lexicographic ordering on B^*, which we also denote $\prec$. We extend $\prec$ to $\mathbf{1} + B^*$ by defining $w \prec \star$ for all $w \in B^*$. Finally, we can lift $\prec$ to the corresponding lexicographic ordering on $(\mathbf{1} + B^*)^{k+1}$.

We now analyse the complexity of computing P_0^S. An insertion in a balanced binary search tree with n nodes can be done in time $O(C \log(n))$, where C is an upper bound for the cost of comparing keys. In our case, the key size is bounded by $|A|\|o\| + \|r\|$, so each insertion can thus be done in time $O(|A|\|o\| + \|r\|) \log(|Q|))$. We have $|Q|$ elements to insert, hence the tree T can be constructed in time $O((|A|\|o\| + \|r\|)|Q| \log(|Q|))$. □

It is difficult to give a compact description of the overall time complexity of carrying out minimisation via normalisation due to the many different factors involved. So in the next proposition we just provide the upper bounds for the two main components of the algorithm. In particular, we do not include the cost of actually constructing the quotient structure.

Proposition 4.15. *Let $\mathbb{S} = (Q, o, d, r)$ be a finite subsequential structure. The time complexities of normalising $\mathbb{S}$ and computing $\mathcal{S}$-bisimilarity on $\mathbb{S}$ are:*

Compute $\mathcal{N}(\mathbb{S})$:	$O((\|\hat{\beta}\| + 1)M)$,
Compute $\mathcal{S}$-bisimilarity on $\mathcal{N}(\mathbb{S})$:	$O((\|A\|\,\|o\| + \|r\|)\|Q\| \log(\|Q\|))$,

where $\|\hat{\beta}\| = \max\{|\hat{\beta}(q)| \mid q \in Q\}$ and M is the number of transitions in $\mathbb{S}$.

Proof. For the normalisation part, we refer to the complexity result given by Béal and Carton [4]. To compute $\mathcal{S}$-bisimilarity on $\mathcal{N}(\mathbb{S})$, we need $O((|A|\,\|o\| + \|r\|)|Q| \log(|Q|))$ time for computing P_0^S (Lemma 4.14), and $O(|A||Q| \log(|Q|))$ time for completing the refinement stage. Since $O(|A||Q| \log(|Q|))$ is dominated by $O((|A|\,\|o\| + \|r\|)|Q| \log(|Q|))$ this gives us the claimed upper bound for computing $\mathcal{S}$-bisimilarity. □

Remark 4.16. A subsequential structure $\mathbb{S} = (Q, o, d, r)$ is *sequential* if $\mathrm{dom}(r) = Q$ and for all $q \in Q$: $r(q) = \varepsilon$. A subsequential transducer $\mathbb{T} = (\mathbb{S}, i, m)$ is a *sequential transducer* if $\mathbb{S}$ is sequential and $m = \varepsilon$. Note that sequential structures/transducers are normalised. The full subcategory Seq of NSubseq with sequential structures as objects is easily seen to be isomorphic to $\mathsf{Coalg}(\mathcal{S}_0)$ where $\mathcal{S}_0(X) = (\mathbf{1} + B^* \times X)^A$. Eilenberg [12] gives a detailed treatment of sequential transducers under the name *generalised sequential machines*. In particular, in [12, Chapter XII] the existence of a final sequential structure is proved, although the words 'final' and 'coalgebra' are never mentioned. The final sequential structure Ω_{Seq} can be characterised as the subcoalgebra of Ω in $\mathsf{Coalg}(\mathcal{S})$ carried by the set of prefix-preserving functions $f : A^* \dashrightarrow B^*$ satisfying $f(\varepsilon) = \varepsilon$.

Similarly, the final Mealy-coalgebra (cf. Example 2.2) can be characterised as the subcoalgebra of Ω_{Seq} in $\mathsf{Coalg}(\mathcal{S})$ whose states are those total maps $f : A^* \to B^*$ in Ω_{Seq} that are also length-preserving.

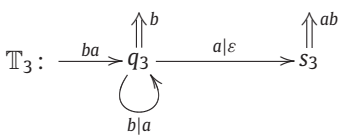
5. Coalgebraisation via differentials

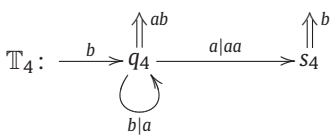
The reason why subsequential structures, in general, cannot be seen as coalgebras essentially comes down to the fact that their semantics allows for asynchrony at internal computation steps, whereas the coalgebraic notion of equivalence requires synchrony at all steps. We have seen that normalisation is one way of eliminating internal asynchrony. In this section, we will see that there is an alternative coalgebraic representation of the class of subsequential structures which have no internal states, and therefore also no proper internal computations. We call this subclass *step-by-step structures*. The coalgebraic representation is obtained by generalising the differential of sequential functions (cf. [12]) to word functions with prefix-closed domain. Taking differential is thus also a form of coalgebraisation, and the differential representation gives rise to an alternative method for determining equivalence.

5.1. Step-by-step structures

Definition 5.1. A subsequential structure $\mathbb{S} = (Q, o, d, r)$ is called *step-by-step* if $\mathrm{dom}(r) = Q$ (i.e. all states are final). A subsequential transducer $(\mathbb{S}, i, m)$ is step-by-step if $\mathbb{S}$ is step-by-step.

Example 5.2. Consider the following two simple step-by-step subsequential transducers.





The behaviour of both $\mathbb{T}_3$ and $\mathbb{T}_4$ is the partial function $f\colon \{a,b\}^* \dashrightarrow \{a,b\}^*$ with $\mathrm{dom}(f) = \{b^k, b^k a \mid k \in \omega\}$ where $f(b^k) = ba^{k+1}b$ and $f(b^k a) = ba^{k+2}b$ for $k \in \omega$.

As we have seen in the above example, step-by-step subsequential transducers are not necessarily normalised, hence two step-by-step subsequential transducers can realise the same function without being in perfect synchrony. Nevertheless, their morphisms can be characterised without explicit reference to an output shift function β.

Proposition 5.3. *Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be step-by-step subsequential structures. A function $\alpha : Q_1 \to Q_2$ is a subsequential morphism if and only if for all $q \in Q_1$ the following hold:*

$$
\begin{aligned}
&(\mathrm{supp}) && \mathrm{supp}(q) = \mathrm{supp}(\alpha(q)),\\
&(\mathrm{next})_{\mathsf{C}} && \forall a \in \mathrm{supp}(q)\colon \alpha(d_1(q)(a)) = d_2(\alpha(q))(a),\\
&(\mathrm{out})_{\mathsf{S}} && \forall a \in \mathrm{supp}(q)\colon\\
& && \quad \overline{r_1(q)} \cdot o_1(q)(a) \cdot r_1(d_1(q)(a)) = \overline{r_2(\alpha(q))} \cdot o_2(\alpha(q))(a) \cdot r_2(d_2(\alpha(q))(a)),\\
&(\text{term-out})_{\mathsf{S}} && r_1(q) \cdot \overline{r_2(\alpha(q))} \in B^*.
\end{aligned}
$$

Let $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_1)$ and $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ be step-by-step subsequential transducers. A function $\alpha : Q_1 \to Q_2$ is a subsequential (transducer) morphism from $\mathbb{T}_1$ to $\mathbb{T}_2$ if and only if $\alpha : \mathbb{S}_1 \to \mathbb{S}_2$ is a subsequential morphism and

$$
\begin{aligned}
&(\mathrm{init}) && \alpha(i_1) = i_2,\\
&(\varepsilon\text{-in})_{\mathsf{S}} && m_1 \cdot r_1(i_1) = m_2 \cdot r_2(i_2).
\end{aligned}
$$

Proof. First assume $\alpha\colon \mathbb{S}_1 \to \mathbb{S}_2$ is a subsequential morphism with witnessing function $\beta\colon Q_1 \to B^*$. Note that since step-by-step structures are coaccessible, α satisfies the conditions from Proposition 3.16, in particular, α must be a total function. Condition (term-out) implies that for all $q \in Q_1$:

$$\beta(q) = r_1(q) \cdot \overline{r_2(\alpha(q))}. \tag{8}$$

Hence, as $\beta(q) \in B^*$, $(\text{term-out})_{\mathsf{S}}$ must hold. Using (8), one can also easily verify that $(\mathrm{out})_{\mathsf{C}}$ reduces to $(\mathrm{out})_{\mathsf{S}}$.

Conversely, for any total function $\alpha\colon Q_1 \to Q_2$ which satisfies the above requirements, we can define $\beta\colon Q \to B^*$ using (8), since condition $(\text{term-out})_{\mathsf{S}}$ guarantees that $\beta(q) \in B^*$. It is now straightforward to verify that $(\alpha, \beta)\colon \mathbb{S}_1 \to \mathbb{S}_2$.

Finally, a subsequential transducer morphism $\alpha\colon \mathbb{T}_1 \to \mathbb{T}_2$ between step-by-step transducers satisfies $(\varepsilon\text{-in})_{\mathsf{S}}$ due to (8) and $(\varepsilon\text{-in})$. Conversely, if $\alpha\colon \mathbb{S}_1 \to \mathbb{S}_2$ is a subsequential morphism satisfying $(\varepsilon\text{-in})_{\mathsf{S}}$, then $(\varepsilon\text{-in})$ must hold for the unique witnessing function β given in (8). □

Let Step and StepTra denote the full subcategories of CSubseq and StepTra which have step-by-step subsequential structures and transducers as their objects, respectively.

The behaviours of step-by-step transducers do not preserve prefixes, as illustrated in Example 5.2. However, since all states are final, the domain is prefix-closed.

Proposition 5.4. *A function $f\colon A^* \dashrightarrow B^*$ is realised by a step-by-step subsequential transducer if and only if* $\mathrm{dom}(f)$ *is prefix-closed.*

Proof. Clearly, if f is realised by a step-by-step subsequential transducer, then $\mathrm{dom}(f)$ is prefix-closed. To prove the other direction, it suffices to show that the minimal realisation $\mathbb{T}_f$ is step-by-step, that is, for all $w \in A^*$, if $w \in \mathrm{dom}(f)$ then $\varepsilon \in \mathrm{dom}(f \cdot w)$. But this is immediate from the definition of $f \cdot w$, cf. Definition 4.5. □

5.2. Differential representations

Although step-by-step behaviours do not preserve prefixes, they have a property which generalises the following basic decomposition property of prefix-preserving functions. If $f\colon A^* \dashrightarrow B^*$ is prefix-preserving then for all $w = a_1 a_2 \ldots a_n \in A^*$, $f(w)$ factors as:

$$f(w) = f(\varepsilon) \cdot f(a_1) \cdot (f \cdot a_1)(a_2) \cdot (f \cdot a_1 a_2)(a_3) \cdot \ldots \cdot (f \cdot a_1 a_2 \ldots a_{n-1})(a_n). \tag{9}$$

The differential of a prefix-preserving f describes the growth of f and is formally defined as the map $D_f\colon A^+ \dashrightarrow B^*$ which for all $wa \in \mathrm{dom}(f)$ is determined by the equation $f(wa) = f(w) \cdot D_f(wa)$ (cf. [12]), that is, $D_f(wa) = \overline{f(w)} \cdot f(wa)$.

It can easily be checked that for a prefix-preserving f, $f[w] = f(w)$, and hence $D_f(wa) = (f{\cdot}w)(a)$. We can now rewrite (9) as:

$$f(w) = f(\varepsilon) \cdot D_f(a_1) \cdot D_f(a_1a_2) \cdot \ldots \cdot D_f(a_1a_2 \ldots a_n). \tag{10}$$

If f does not preserve prefixes, we may not be able to decompose f-values as in (9). For example, if $f(a_1)$ is not a prefix of $f(a_1a_2)$, then $f(a_1a_2) \neq f(a_1) \cdot (f{\cdot}a_1)(a_2)$. However, if f has a prefix-closed domain, f can still be decomposed using the differential by allowing D_f to take values in the free group $B^{(*)}$ rather than B^*. This generalisation of the differential was introduced in [32].

Definition 5.5. Let $f : A^* \dashrightarrow B^*$ be a function with prefix-closed domain. The *differential of* f is the partial function $D_f : A^+ \dashrightarrow B^{(*)}$ defined on $\mathrm{dom}(f) \setminus \{\varepsilon\}$ for all $a \in A$, $w \in A^*$ by

$$D_f(wa) = \overline{f(w)} \cdot f(wa).$$

Lemma 5.6. *Let $f : A^* \dashrightarrow B^*$ be a function with prefix-closed domain. For all $w = a_1a_2 \ldots a_n \in \mathrm{dom}(f)$, $n \geq 1$, we have:*

$$f(w) = f(\varepsilon) \cdot D_f(a_1) \cdot D_f(a_1a_2) \cdot \ldots \cdot D_f(a_1a_2 \ldots a_n), \tag{11}$$
$$D_f(w) = D_{f\cdot a_1 \ldots a_{n-1}}(a_n), \tag{12}$$
$$f(w) = f(\varepsilon) \cdot D_{f\cdot\varepsilon}(a_1) \cdot D_{f\cdot a_1}(a_2) \cdot D_{f\cdot a_1a_2}(a_3) \cdot \ldots \cdot D_{f\cdot a_1\ldots a_{n-1}}(a_n). \tag{13}$$

Proof. Eq. (11) holds more or less by definition of D_f:

$$\begin{aligned} f(w) &= f(\varepsilon) \cdot \overline{f(\varepsilon)} \cdot f(a_1) \cdot \overline{f(a_1)} \cdot f(a_1a_2) \cdot \ldots \cdot \overline{f(a_1a_2 \ldots a_{n-1})} \cdot f(a_1a_2 \ldots a_n) \\ &= f(\varepsilon) \cdot D_f(a_1) \cdot D_f(a_1a_2) \cdot \ldots \cdot D_f(a_1a_2 \ldots a_n). \end{aligned}$$

To see that Eq. (12) holds, let $v = a_1 \ldots a_{n-1}$. We have:

$$D_{f\cdot v}(a_n) = \overline{(f{\cdot}v)(\varepsilon)} \cdot (f{\cdot}v)(a_n) = \overline{\overline{f[v]} \cdot f(v)} \cdot \overline{f[v]} f(va_n) = \overline{f(v)} \cdot f(va_n) = D_f(va_n).$$

Eq. (13) follows from (11) and (12). □

Example 5.7. We compute the differential of the function f realised by the step-by-step subsequential transducers from Example 5.2. Recall that $f : \{a, b\}^* \dashrightarrow \{a, b\}^*$ with $\mathrm{dom}(f) = b^* \cup b^*a$ where $f(b^k) = ba^{k+1}b$ and $f(b^ka) = ba^{k+2}b$ for $k \in \omega$. We have for $k \geq 1$:

$$\begin{aligned} D_f(b^k) &= \overline{f(b^{k-1})} \cdot f(b^k) = \overline{ba^kb} \cdot ba^{k+1}b = \bar{b}ab \quad \text{for } k \geq 1, \\ D_f(b^ka) &= \overline{f(b^k)} \cdot f(b^ka) = \overline{ba^{k+1}b} \cdot ba^{k+2}b = \bar{b}ab \quad \text{for } k \geq 0. \end{aligned}$$

The analogue of (12) for differentials of state behaviour are shown in the following lemma. Note that this lemma is not an immediate consequence of (12), since $[\![q]\!]{\cdot}w \neq [\![d(q)(w)]\!]$ if $d(q)(w)$ is not normalised which can be the case in a step-by-step structure.

Lemma 5.8. *Let $\mathbb{S} = (Q, o, d, r)$ be a step-by-step subsequential structure, $q_0 \in Q$ and $w = a_1 \ldots a_n \in \mathrm{dom}([\![q_0]\!])$, $n \geq 1$. For ease of notation, let $q_k = d(q_0)(a_1 \ldots a_k)$ and $u_k = o(q_{k-1})(a_k)$ for $k = 1, \ldots, n$, and let $r_k = r(q_k)$ for $k = 0, \ldots, n$, as illustrated in the following picture:*

$$\begin{array}{ccccccc} \uparrow r_0 & & \uparrow r_1 & & \uparrow r_2 & & \\ q_0 & \xrightarrow{a_1|u_1} & q_1 & \xrightarrow{a_2|u_2} & q_2 & \cdots & \\ & & & & & & \\ & & & & \uparrow r_{n-1} & & \uparrow r_n \\ & & & & q_{n-1} & \xrightarrow{a_n|u_n} & q_n \end{array}$$

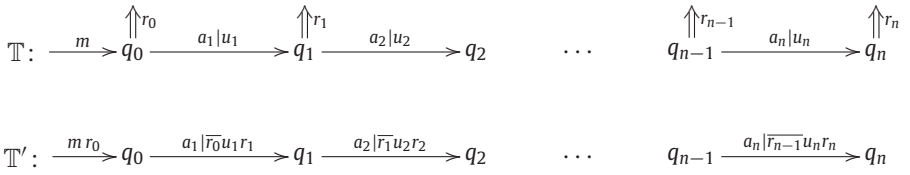
We have for all $k \in \{1, \ldots, n\}$: $D_{[\![q_0]\!]}(a_1, \ldots, a_k) = D_{[\![q_{k-1}]\!]}(a_k)$.

Proof. For $k \in \{1, \ldots, n\}$ we have by definition:

$$\begin{aligned} D_{[\![q_0]\!]}(a_1 \ldots a_k) &= \overline{[\![q_0]\!](a_1 \ldots a_{k-1})} \cdot [\![q_0]\!](a_1 \ldots a_k) \\ &= \overline{u_1 \cdot \ldots \cdot u_{k-1} \cdot r_{k-1}} \cdot u_1 \cdot \ldots \cdot u_{k-1} \cdot u_k \cdot r_k \end{aligned}$$

$$= \overline{r_{k-1}} \cdot u_k \cdot r_k$$
$$= \overline{[\![q_{k-1}]\!](\varepsilon)} \cdot [\![q_{k-1}]\!](a_k)$$
$$= D_{[\![q_{k-1}]\!]}(a_k). \quad \square$$

Representing behaviour in terms of the differential can be seen as transforming a step-by-step transducer $\mathbb{T}$ into a sequential transducer $\mathbb{T}'$ with prefix which produces output in the free group $B^{(*)}$. A computation in $\mathbb{T}$ corresponds to a computation in $\mathbb{T}'$ as illustrated here:



Recall from Remark 4.16 that a subsequential structure is sequential, if the terminal output function r is constant equal to ε. We will therefore often leave out r from the specification of a sequential structure. Sequential structures with output in $B^{(*)}$ are not essentially different from sequential structures with output in B^*, and all previously introduced notions for sequential structures apply with identity taken in $B^{(*)}$ where appropriate. This includes extending the transition output function from letters to words, and the definitions of sequential morphisms and behaviour. Differential representations of transducers are almost sequential transducers with output in $B^{(*)}$. The only difference is that differential representations may have a non-trivial initial prefix, whereas sequential transducers have empty initial prefix by definition.

Definition 5.9. We denote by $\mathsf{Seq}^{(*)}$ the category of sequential structures $\mathbb{S} = (Q, o, d)$ in which the transition output function may take values in $B^{(*)}$, i.e., $o\colon Q \to (A \dashrightarrow B^{(*)})$. The morphisms of $\mathsf{Seq}^{(*)}$ are the functions which satisfy the conditions (supp), (next) and $(\mathrm{out})_{\mathsf{N}}$ by taking equality in $B^{(*)}$ in $(\mathrm{out})_{\mathsf{N}}$.

We denote by $\mathsf{pSeqTra}^{(*)}$ the category which has as its objects subsequential transducers $\mathbb{T} = (\mathbb{S}, i, m)$, where $\mathbb{S}$ is an object in $\mathsf{Seq}^{(*)}$. A morphism in $\mathsf{pSeqTra}^{(*)}$ is a $\mathsf{Seq}^{(*)}$-morphism of the underlying structures which maps initial state to initial state and leaves the initial prefix unchanged.

Remark 5.10. The p in $\mathsf{pSeqTra}^{(*)}$ is used to indicate that the objects can have a non-trivial prefix, since the name $\mathsf{SeqTra}^{(*)}$ suggests objects that are sequential transducers which by definition have empty prefix.

The following definition makes the transformation suggested after Lemma 5.8 precise.

Definition 5.11 (*differential representation*)**.** Let $\mathbb{S} = (Q, o, d, r)$ be a step-by-step subsequential structure. The *differential representation of* $\mathbb{S}$ is the object in $\mathsf{Seq}^{(*)}$ denoted by $\mathcal{D}(\mathbb{S}) = (Q, \partial_{\mathbb{S}}, d)$, where the output function $\partial_{\mathbb{S}}\colon Q \to (\mathbf{1} + B^{(*)})^A$ is defined for $q \in Q$ and $a \in A$ by:

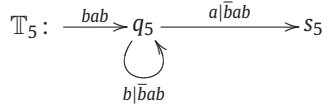
$$\partial_{\mathbb{S}}(q)(a) = \overline{r(q)} \cdot o(q)(a) \cdot r(d(q)(a)) \tag{14}$$

if $a \in \mathrm{supp}(q)$, and $\bigstar$ otherwise.

For a step-by-step subsequential transducer $\mathbb{T} = (\mathbb{S}, i, m)$ where $\mathbb{S} = (Q, o, d, r)$, we define the *differential representation of* $\mathbb{T}$ as the object in $\mathsf{pSeqTra}^{(*)}$ defined by $\mathcal{D}(\mathbb{T}) = (\mathcal{D}(\mathbb{S}), i, m \cdot r(i))$.

As with other subscripts, we may leave out $\mathbb{S}$ from $\partial_{\mathbb{S}}$ and simply write ∂ if $\mathbb{S}$ is immaterial, or clear from the context. When we speak of the differential representation of a structure $\mathbb{S}$ or a transducer $\mathbb{T}$, we will always implicitly assume that $\mathbb{S}$ and $\mathbb{T}$ are step-by-step.

Example 5.12. It is straightforward to check that the differential representations of the two step-by-step subsequential transducers $\mathbb{T}_3$ and $\mathbb{T}_4$ from Example 5.2 are both isomorphic to the object in $\mathsf{pSeqTra}^{(*)}$ depicted below. For example, in $\mathbb{T}_4$, the differential output function at q_4 in b is: $\partial(q_4)(b) = \overline{ab} \cdot a \cdot ab = \bar{b}ab$.



Taking differential representations of structures is a map from the objects of Step to the objects of $\mathsf{Seq}^{(*)}$, and as with normalisation, we would like to give a formal argument in the form of a reflectivity result which says that the right way of looking at step-by-step structures is in terms of their differential representation. However, at first glance there is a problem, since $\mathsf{Seq}^{(*)}$ is not a subcategory of Step. The solution to this problem is to also generalise step-by-step structures to produce output in $B^{(*)}$.

Definition 5.13. The category $\mathsf{Step}^{(*)}$ has as its objects step-by-step structures $\mathbb{S} = (Q, o, d, r)$ in which the transition output function may take values in $B^{(*)}$, i.e., Q, d and r are as in Definition 5.1 and $o\colon Q \to (A \dashrightarrow B^{(*)})$. The morphisms of $\mathsf{Step}^{(*)}$ are the functions which satisfy the characterising conditions for Step-morphism given in Proposition 5.3.

The category $\mathsf{StepTra}^{(*)}$ consists of subsequential transducer objects $\mathbb{T} = (\mathbb{S}, i, m)$ where $\mathbb{S}$ is in $\mathsf{Step}^{(*)}$ together with the functions which satisfy the characterising conditions for StepTra-morphisms given in Proposition 5.3.

Again, previously defined notions and results, including behaviour and differential representations, all apply unchanged to $\mathsf{Step}^{(*)}$ and $\mathsf{StepTra}^{(*)}$. From the definition of $\mathsf{Seq}^{(*)}$ and $\mathsf{Step}^{(*)}$, and the natural embedding of B^* in $B^{(*)}$, it is clear that $\mathsf{Seq}^{(*)}$ and Step are full subcategories of $\mathsf{Step}^{(*)}$, and $\mathsf{pSeqTra}^{(*)}$ and StepTra are full subcategories of $\mathsf{StepTra}^{(*)}$.

We now have a suitable set-up of subcategories, and $\mathcal{D}$ defines an object map from $\mathsf{Step}^{(*)}$ to $\mathsf{Seq}^{(*)}$ and from $\mathsf{StepTra}^{(*)}$ to $\mathsf{pSeqTra}^{(*)}$.

Theorem 5.14. *Let $\mathbb{S} \in \mathsf{Step}^{(*)}$ and $\mathbb{T} \in \mathsf{StepTra}^{(*)}$. We have:*

1. *$id_{\mathbb{S}}\colon \mathbb{S} \to \mathcal{D}(\mathbb{S})$ is a $\mathsf{Seq}^{(*)}$-reflection arrow for $\mathbb{S}$, and*
2. *$id_{\mathbb{T}}\colon \mathbb{T} \to \mathcal{D}(\mathbb{T})$ is a $\mathsf{pSeqTra}^{(*)}$-reflection arrow for $\mathbb{T}$.*

Hence $\mathsf{Seq}^{(*)}$ is a reflective subcategory of $\mathsf{Step}^{(*)}$, and $\mathsf{pSeqTra}^{(*)}$ is a reflective subcategory of $\mathsf{StepTra}^{(*)}$. Moreover, by defining $\mathcal{D}(\alpha) = \alpha$ for all morphisms α in $\mathsf{Step}^{(*)}$ and $\mathsf{StepTra}^{(*)}$, $\mathcal{D}$ is a functor $\mathcal{D}\colon \mathsf{Step}^{(*)} \to \mathsf{Seq}^{(*)}$, and $\mathcal{D}\colon \mathsf{StepTra}^{(*)} \to \mathsf{pSeqTra}^{(*)}$.

Proof. Let $\mathbb{S} = (Q, o, d, r)$ be an object in $\mathsf{Step}^{(*)}$. We first check that $id_{\mathbb{S}}$ is a $\mathsf{Step}^{(*)}$-morphism from $\mathbb{S}$ to $\mathcal{D}(\mathbb{S})$, that is, $id_{\mathbb{S}} = id_Q$ satisfies the conditions given in Proposition 5.3 when taking identity in $B^{(*)}$ in $(\mathrm{out})_{\mathsf{S}}$. The conditions (supp) and $(\mathrm{next})_{\mathsf{C}}$ clearly hold for id_Q. In $\mathcal{D}(\mathbb{S})$ the terminal output function is constant equal to ε, hence $(\mathrm{out})_{\mathsf{S}}$ reduces to the requirement that for all $q \in Q$ and $a \in \mathrm{supp}(q)$: $\overline{r(q)} \cdot o(q)(a) \cdot r(d(q)(a)) = \partial_{\mathbb{S}}(q)(a)$. This is just the definition of $\partial_{\mathbb{S}}$, hence true. Similarly, condition $(\text{term-out})_{\mathsf{S}}$ reduces to $r(q) \in B^*$ for all $q \in Q$, which also clearly holds.

We must now prove that for any $\mathbb{S}' = (Q', o', d') \in \mathsf{Seq}^{(*)}$ and $\alpha\colon \mathbb{S} \to \mathbb{S}'$ in $\mathsf{Step}^{(*)}$, there is a unique $\mathsf{Seq}^{(*)}$-morphism $\alpha'\colon \mathcal{D}(\mathbb{S}) \to \mathbb{S}'$ such that $\alpha = \alpha' \circ id_Q$. As when showing that normalisation is a reflector (Theorem 3.31), we will prove that $\alpha' = \alpha$ is the unique choice. If α is a $\mathsf{Seq}^{(*)}$-morphism from $\mathcal{D}(\mathbb{S})$ to $\mathbb{S}'$, then α is clearly the unique morphism such that $\alpha = \alpha \circ id_Q$. To prove that $\alpha\colon \mathcal{D}(\mathbb{S}) \to \mathbb{S}'$ we first note that (supp) and $(\mathrm{next})_{\mathsf{N}}$ are satisfied since the underlying DAs of $\mathcal{D}(\mathbb{S})$ and $\mathbb{S}$ are identical. By the assumption that α is a $\mathsf{Step}^{(*)}$-morphism, α satisfies $(\mathrm{out})_{\mathsf{S}}$, i.e., for all $q \in Q$ and $a \in \mathrm{supp}(q)$: $\partial_{\mathbb{S}}(q)(a) = o'(q)(a)$, hence $(\mathrm{out})_{\mathsf{N}}$ holds and $\alpha\colon \mathcal{D}(\mathbb{S}) \to \mathbb{S}'$.

We leave it to the reader to extend the proof for $\mathsf{Step}^{(*)}$ to $\mathsf{pStepTra}^{(*)}$. □

Corollary 5.15. *For any $\mathbb{T}$ in StepTra, we have: $[\![\mathbb{T}]\!] = [\![\mathcal{D}(\mathbb{T})]\!]$.*

Proof. This is an immediate consequence of Theorem 5.14 and the behaviour preservation of subsequential morphisms (which also holds in $\mathsf{StepTra}^{(*)}$). □

Going to the differential representation only preserves state behaviour modulo an output shift, since differential structures are not normalised.

Lemma 5.16. *Let $\mathbb{S} = (Q, o, d, r)$ be a step-by-step structure, and $q \in Q$. We have: $[\![q]\!]_{\mathbb{S}} = r(q) \cdot [\![q]\!]_{\mathcal{D}(\mathbb{S})}$.*

Proof. The implicitly defined output shift function for the reflection arrow $id_{\mathbb{S}}\colon \mathbb{S} \to \mathcal{D}(\mathbb{S})$ is $\beta = r$ (cf. equation (8)). The result now follows from Proposition 3.9. □

Although state behaviour is not preserved, equivalence in the differential structure captures equivalence of differentials.

Proposition 5.17. *Let $\mathbb{S} = (Q, o, d, r)$ be a step-by-step structure, and $q_1, q_2 \in Q$. We have: $D_{[\![q_1]\!]_{\mathbb{S}}} = D_{[\![q_2]\!]_{\mathbb{S}}}$ iff $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}$.*

Proof. Since $\mathcal{D}(\mathbb{S})$ is a sequential structure, we always have $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})}(\varepsilon) = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}(\varepsilon) = \varepsilon$. For $w = a_1 \dots a_n \in A^+$, $n \geq 1$, and any $q \in Q$ we have

$$\begin{aligned}
[\![q]\!]_{\mathcal{D}(\mathbb{S})}(w) &= \partial_{\mathbb{S}}(q)(a_1) \centerdot \partial_{\mathbb{S}}(d(q)(a_1))(a_2) \centerdot \ \dots \ \centerdot \partial_{\mathbb{S}}(d(q)(a_1 \dots a_{n-1}))(a_n) \\
&= D_{[\![q]\!]}(a_1) \centerdot D_{[\![d(q)(a_1)]\!]}(a_2) \centerdot \ \dots \ \centerdot D_{[\![d(q)(a_1 \dots a_{n-1})]\!]}(a_n) \\
\text{(Lemma 5.8)} &= D_{[\![q]\!]}(a_1) \centerdot D_{[\![q]\!]}(a_1 a_2) \centerdot \ \dots \ \centerdot D_{[\![q]\!]}(a_1 \dots a_n).
\end{aligned}$$

It follows from the above that $D_{[\![q_1]\!]} = D_{[\![q_2]\!]}$ if and only if $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}$. □

5.3. Coalgebras for differentials

Objects from $\mathsf{Seq}^{(*)}$ can be modelled as coalgebras in the same way as sequential structures (cf. Remark 4.16) by changing the type functor accordingly. Let the functor $\mathcal{S}_0^{(*)} : \mathsf{Set} \to \mathsf{Set}$ be defined by:

$$\begin{aligned}
\mathcal{S}_0^{(*)}(X) &= (\mathbf{1} + B^{(*)} \times X)^A, \\
\mathcal{S}_0^{(*)}(f : X \to Y) &= (\mathbf{1} + id_{B^{(*)}} \times f)^{id_A}.
\end{aligned} \tag{15}$$

Proposition 5.18. $\mathsf{Seq}^{(*)}$ *is isomorphic to* $\mathsf{Coalg}(\mathcal{S}_0^{(*)})$.

Proof. Any structure $\mathbb{S} = (Q, o, d)$ in $\mathsf{Seq}^{(*)}$ can be seen as an $\mathcal{S}_0^{(*)}$-coalgebra:

$$\langle \partial_{\mathbb{S}}, d \rangle : Q \to (\mathbf{1} + B^{(*)} \times Q)^A.$$

Checking that the morphisms of $\mathsf{Seq}^{(*)}$ and $\mathsf{Coalg}(\mathcal{S}_0^{(*)})$ coincide is more or less immediate from the definition of $\mathsf{Seq}^{(*)}$-morphisms. □

The existence of a final object in $\mathsf{Seq}^{(*)}$ does not follow from the existence of a final sequential structure (Remark 4.16), but since $S_0^{(*)}$ is a polynomial functor, we know that such a final object exists, and we will see it is straightforward to prove this from first principles. As expected, the final object will have the state behaviours of structures in $\mathsf{Seq}^{(*)}$ as its carrier. Now we adjust the definition of derivative to functions with codomain $B^{(*)}$ and prefix-closed domains (as e.g. differentials). Let $f : A^* \dashrightarrow B^{(*)}$ be a function with prefix-closed domain, and let $a \in A$. The derivative of f with respect to a is the partial function $f{\cdot}a : A^* \dashrightarrow B^{(*)}$ defined for all $w \in A^*$ by $(f{\cdot}a)(w) = \overline{f(a)} \centerdot f(aw)$ if $aw \in \mathrm{dom}(f)$.

Theorem 5.19. *Define*

$$Q_{\mathsf{Seq}^{(*)}} = \{f : A^* \dashrightarrow B^{(*)} \mid \mathrm{dom}(f) \text{ is prefix-closed}, f(\varepsilon) = \varepsilon\}.$$

For $f \in Q_{\mathsf{Seq}^{(*)}}$ *and* $a \in A$, *define*

$$O^{(*)}(f)(a) = f(a), \qquad D^{(*)}(f)(a) = f{\cdot}a, \qquad R^{(*)}(f) = f(\varepsilon).$$

The 4-triple $\Omega_{\mathsf{Seq}^{(*)}} = (Q_{\mathsf{Seq}^{(*)}}, O^{(*)}, D^{(*)}, R^{(*)})$ *is a final object in* $\mathsf{Seq}^{(*)}$, *and for all* $\mathbb{S} \in \mathsf{Seq}^{(*)}$, *the final morphism is* $[\![\_]\!]_{\mathbb{S}}$.

Proof. We first check that $\Omega_{\mathsf{Seq}^{(*)}}$ is well-defined. The output function $O^{(*)}$ takes values in $B^{(*)}$ and the terminal output function $R^{(*)}$ is constant equal to ε on $Q_{\mathsf{Seq}^{(*)}}$, hence if $Q_{\mathsf{Seq}^{(*)}}$ is closed under taking derivatives, then we can conclude that $\Omega_{\mathsf{Seq}^{(*)}}$ is a well-defined object in $\mathsf{Seq}^{(*)}$. Let $f \in Q_{\mathsf{Seq}^{(*)}}$ and $a \in A$. We have $w \in \mathrm{dom}(f{\cdot}a)$ iff $aw \in \mathrm{dom}(f)$, so by the assumption that $\mathrm{dom}(f)$ is prefix-closed, it follows that $\mathrm{dom}(f{\cdot}a)$ is prefix-closed. Moreover, $(f{\cdot}a)(\varepsilon) = \overline{f(a)} \centerdot f(a \centerdot \varepsilon) = \varepsilon$.

We now show that the behaviour map $[\![\_]\!]$ is the final map, i.e., for any $\mathbb{S}$ in $\mathsf{Seq}^{(*)}$, $[\![\_]\!] : \mathbb{S} \to \Omega_{\mathsf{Seq}^{(*)}}$ is the unique $\mathsf{Seq}^{(*)}$-morphism. Let $\mathbb{S} = (Q, o, d, r)$ be a sequential structure in $\mathsf{Seq}^{(*)}$. First of all, since all states in $\mathbb{S}$ are final, it is clear that for any $q \in Q$, $\mathrm{dom}([\![q]\!])$ is prefix-closed, hence $[\![q]\!] \in Q_{\mathsf{Seq}^{(*)}}$. The condition (supp) is easily seen to hold, namely, for $q \in Q$ and $a \in A$ we have: $a \in \mathrm{supp}(q)$ iff $a \in \mathrm{dom}([\![q]\!])$ iff $a \in \mathrm{supp}([\![q]\!])$. Also immediate is the condition $(\mathrm{out})_{\mathsf{N}}$, since for all $q \in Q$ and $a \in \mathrm{supp}(q)$, $[\![q]\!](a) = o(q)(a)$ by definition. Finally, to see that $(\mathrm{next})_{\mathsf{C}}$ holds, we have for all $q \in Q$, $a \in \mathrm{supp}(q)$ and $w \in A^*$:

$$[\![d(q)(a)]\!](w) = \overline{o(q)(a)} \centerdot [\![q]\!](aw) = ([\![q]\!]{\cdot}a)(w).$$

We have thus shown that for any $\mathbb{S}$ in $\mathsf{Seq}^{(*)}$, the map $[\![\_]\!]\colon \mathbb{S} \to \Omega_{\mathsf{Seq}^{(*)}}$ is a $\mathsf{Seq}^{(*)}$-morphism. We leave uniqueness as an exercise to the reader. □

Since $\mathsf{Seq}^{(*)}$ is reflective in $\mathsf{Step}^{(*)}$ it follows that $\Omega_{\mathsf{Seq}^{(*)}}$ is also a final object in $\mathsf{Step}^{(*)}$. Hence for any step-by-step $\mathbb{S}$, considered as an object in $\mathsf{Step}^{(*)}$, there is a unique $\mathsf{Step}^{(*)}$-morphism from $\mathbb{S}$ to $\Omega_{\mathsf{Seq}^{(*)}}$. By combining the reflectivity of $\mathsf{Seq}^{(*)}$ in $\mathsf{Step}^{(*)}$ with the coalgebraic modelling of $\mathsf{Seq}^{(*)}$ we can now argue that the differential representation is an alternative to normalisation which provides an equally correct way of viewing step-by-step structures as coalgebras. From Theorem 5.19 it also follows that state equivalence is bisimilarity in differential representations.

Corollary 5.20. *Let $\mathbb{S}_1$ and $\mathbb{S}_2$ be a step-by-step structures. For all states q_1 in $\mathbb{S}_1$ and q_2 in $\mathbb{S}_2$: $q_1 \sim_{\mathcal{S}_0^{(*)}} q_2$ iff $[\![q_1]\!]_{\mathcal{D}(\mathbb{S}_1)} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S}_2)}$.*

Proof. By Proposition 5.18 and Theorem 5.19, $[\![\_]\!]$ is the final $\mathsf{Coalg}(\mathcal{S}_0^{(*)})$-map, which implies that $q_1 \equiv_{\mathcal{S}_0^{(*)}} q_2$ iff $[\![q_1]\!]_{\mathcal{D}(\mathbb{S}_1)} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S}_2)}$. The result now follows from Proposition 2.3(2). □

5.4. Minimising differential representations

The definition of $\mathcal{S}_0^{(*)}$-bisimulation amounts to the following. Let $\mathbb{S} = (Q, o, d)$ be an $\mathcal{S}_0^{(*)}$-coalgebra. A relation $R \subseteq Q \times Q$ is an $\mathcal{S}_0^{(*)}$-*bisimulation* on $\mathbb{S}$, if for all $\langle q, s\rangle \in R$:

(d0) $\mathrm{supp}(q) = \mathrm{supp}(s)$;
(d1) for all $a \in \mathrm{supp}(q)$: $o(q)(a) = o(s)(a)$ (in the free group $B^{(*)}$); and
(d2) for all $a \in \mathrm{supp}(q)$: $\langle d(q)(a), d(s)(a)\rangle \in R$.

Given a finite $\mathcal{S}_0^{(*)}$-coalgebra $\mathbb{S} = (Q, o, d)$, we can compute $\mathcal{S}_0^{(*)}$-bisimilarity on $\mathbb{S}$ by using a small variation on the algorithm from Section 4.3 for computing $\mathcal{S}$-bisimilarity in normalised structures. The idea is again to perform the refinement algorithm for DFAs, but this time we take as the initial partition, the largest equivalence relation P_0^d on Q which satisfies (d0) and (d1).

Lemma 5.21. *Let $\mathbb{S} = (Q, o, d)$ be a finite $\mathcal{S}_0^{(*)}$-coalgebra. We can compute the largest equivalence relation P_0^d on Q which satisfies (d0) and (d1) in time $O(\|o\|\,|A||Q|\log(|Q|))$ where $\|o\| := \max\{|o(q)(a)| \mid q \in Q, a \in A\}$.*

Proof. P_0^d can be computed in essentially the same way as P_0^s, so we only provide a sketch and refer to Lemma 4.14 for details. We again use a binary search tree, but now a state q is inserted with key value $c(q) := \langle o(q)(a_1), \ldots, o(q)(a_k)\rangle$ where $|A| = k$. In order to define a linear ordering on key values, it suffices to define a linear ordering on $B \cup \{\overline{b} \mid b \in B\}$, since we can then extend this ordering lexicographically to reduced elements of $B^{(*)}$ and key values as in Lemma 4.14. As before, we obtain a linear ordering $b_1 \prec b_2 \prec \ldots \prec b_n$ on $B = \{b_1, b_2, \ldots, b_n\}$ by enumeration, and we extend $\prec$ to $B \cup \{\overline{b} \mid b \in B\}$ by defining $\overline{b} \prec \overline{b'}$ iff $b \prec b'$ and $\overline{b} \prec b'$ for all $b, b' \in B$, i.e., $\overline{b_1} \prec \overline{b_2} \prec \ldots \prec \overline{b_n} \prec b_1 \prec b_2 \prec \ldots \prec b_n$. The size of $c(q)$-values is now $O(|A|\|o\|)$ which yields a time complexity of $O(\|o\|\,|A|\,|Q|\log(|Q|))$ for inserting all pairs $(c(q), q)$ into the tree. □

Lemma 5.21 can be used to give a bound on the complexity of computing $\mathcal{S}_0^{(*)}$-bisimilarity on $\mathcal{D}(\mathbb{S})$ starting with a finite step-by-step structure $\mathbb{S}$.

Proposition 5.22. *Let $\mathbb{S} = (Q, o, d, r)$ be a finite step-by-step structure. The time complexities of computing $\mathcal{S}_0^{(*)}$-bisimilarity on $\mathcal{D}(\mathbb{S})$ are:*

Compute $\partial_{\mathbb{S}}$:	$O(M\|r\|)$,
Compute $\mathcal{S}_0^{(*)}$-bisimilarity on $\mathcal{D}(\mathbb{S})$:	$O((2\|r\| + \|o\|)\,\|A\|\,\|Q\|\log(\|Q\|))$

where M is the number of transitions in $\mathbb{S}$, $\|r\| := \max\{|r(q)| \mid q \in Q\}$ and $\|o\| := \max\{|o(q)(a)| \mid q \in Q, a \in A\}$.

Proof. We must compute $\partial_{\mathbb{S}}(q)(a) = \overline{r(q)} \cdot o(q)(a) \cdot r(d(q)(a))$ for each $q \in Q$ and $a \in \mathrm{supp}(a)$. Since we want to compare $\partial_{\mathbb{S}}$-values, we want to expand $\overline{r(q)}$ to a string of the form $\overline{b_1} \ldots \overline{b_n}$ where $b_i \in B$, $i = 1, \ldots, n$. Hence computing all $\partial_{\mathbb{S}}$-values can be done in time $O(M\|r\|)$. From Lemma 5.21 we know that we can compute the initial partition P_0^d on $\mathcal{D}(\mathbb{S})$ in time $O(\|\partial_{\mathbb{S}}\|\,|A|\,|Q|\log(|Q|))$. From the definition of $\partial_{\mathbb{S}}$, we have that $\|\partial_{\mathbb{S}}\| \leq 2\|r\| + \|o\|$. Hence P_0^d can be computed in time $O((2\|r\| + \|o\|)\,|A|\,|Q|\log(|Q|))$. The refinement part of the algorithm can be done in time $O(|A||Q|\log(|Q|))$

(cf. [23]). Adding up the time needed for computing P_0^d and the time needed for refinement (under the big-O), we find that $\mathcal{S}_0^{(*)}$-bisimilarity can be computed in time $O((2\|r\| + \|o\|)\,|A|\,|Q|\log(|Q|))$. □

Proposition 5.22 gives us a method to decide equivalence of step-by-step transducers via differentials without normalisation.

Theorem 5.23. *Let $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_1)$ and $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ be two step-by-step transducers, where $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1))$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$. We have: $\mathbb{T}_1$ and $\mathbb{T}_2$ are equivalent if and only if*

$$m_1 \centerdot r_1(i_1) = m_2 \centerdot r_2(i_2) \quad \text{and} \quad i_1 \sim_{\mathcal{S}_0^{(*)}} i_2.$$

We can decide whether $[\![\mathbb{T}_1]\!] = [\![\mathbb{T}_2]\!]$ in time: $O(\|m\| + (2\|r\| + \|o\|)|A|N\log(N))$, where $\|m\| = \max\{|m_1|, |m_2|\}$. $N = |Q_1| + |Q_2|$, $\|r\| = \max\{|r_j(q_j)| \mid q_j \in Q_j, j \in \{1,2\}\}$, $\|o\| = \max\{|o_j(q_j)(a)| \mid q_j \in Q_j, j \in \{1,2\}, a \in A\}$.

Proof. From the definition of $\mathcal{D}(\mathbb{T})$ and Corollary 5.15, we have:

$$[\![\mathbb{T}_j]\!] = [\![\mathcal{D}(\mathbb{T}_j)]\!] = m_j \centerdot r_j(i_j) \centerdot [\![i_j]\!]_{\mathcal{D}(\mathbb{S}_j)} \text{ for } j \in \{1,2\}.$$

From Corollary 5.20 it follows that $[\![\mathbb{T}_1]\!] = [\![\mathbb{T}_2]\!]$ iff $m_1 \centerdot r_1(i_1) = m_2 \centerdot r_2(i_2)$ and $i_1 \sim_{\mathcal{S}_0^{(*)}} i_2$. We can determine whether $i_1 \sim_{\mathcal{S}_0^{(*)}} i_2$ by computing $\mathcal{S}_0^{(*)}$-bisimilarity on the coproduct $\mathcal{D}(\mathbb{S}_1) + \mathcal{D}(\mathbb{S}_2)$, cf. Proposition 2.3(v). By Proposition 5.22, we can compute $\partial_{\mathbb{S}_1}$ and $\partial_{\mathbb{S}_2}$, in time $O(M\|r\|)$, where M is the sum of the number of transitions in $\mathbb{S}_1$ and $\mathbb{S}_2$. The coproduct of two $\mathcal{S}_0^{(*)}$-coalgebras is just their disjoint union, hence $\mathcal{S}_0^{(*)}$-bisimilarity on $\mathcal{D}(\mathbb{S}_1) + \mathcal{D}(\mathbb{S}_2)$ can be computed in time $O((2\|r\| + \|o\|)|A|N\log(N))$. Using that $M \leq |A|N$, we can decide $i_1 \sim_{\mathcal{S}_0^{(*)}} i_2$ in time $O((2\|r\| + \|o\|)|A|N\log(N))$. Finally, the equality $m_1 \centerdot r_1(i_1) = m_2 \centerdot r_2(i_2)$, can be checked in time $O(\|m\| + \|r\|)$. Adding up, we find that the time for the entire decision method is $O(\|m\| + (2\|r\| + \|o\|)|A|N\log(N))$. □

We now have two ways of constructing a minimal representation of a step-by-step transducer $\mathbb{T}$. One is quotienting $\mathcal{N}(\mathbb{T})$ with $\mathcal{S}$-bisimilarity, the other is quotienting $\mathcal{D}(\mathbb{T})$ with $\mathcal{S}_0^{(*)}$-bisimilarity. We will show in Proposition 5.25 below that for a step-by-step transducer $\mathbb{T}$, the state equivalence relations on $\mathcal{D}(\mathbb{T})$ and $\mathcal{N}(\mathbb{T})$ are identical (as relations on the state set). Once this is established, it is easy to prove, in Theorem 5.27, that minimising the differential representation $\mathcal{D}(\mathbb{T})$ yields the differential representation of the minimisation of $\mathbb{T}$. First we make an easy, but useful observation.

Lemma 5.24. *If $\mathbb{S}_1, \mathbb{S}_2$ are in* Step*, and $\alpha : \mathbb{S}_1 \to \mathbb{S}_2$ is a subsequential morphism, then for all states q in $\mathbb{S}_1$ and all $a \in A$ we have: $\partial_{\mathbb{S}_1}(q)(a) = \partial_{\mathbb{S}_2}(\alpha(q))(a)$. In particular, since $id_{\mathbb{S}} : \mathbb{S} \to \mathcal{N}(\mathbb{S})$ is a subsequential morphism, $\partial_{\mathbb{S}}(q)(a) = \partial_{\mathcal{N}(\mathbb{S})}(q)(a)$ for all $q \in Q$ and $a \in A$, which implies that $\mathcal{D}(\mathbb{S}) = \mathcal{D}(\mathcal{N}(\mathbb{S}))$.*

Proof. Follows from the fact that $\alpha : \mathbb{S}_1 \to \mathbb{S}_2$ in Step implies that $\alpha : \mathcal{D}(\mathbb{S}_1) \to \mathcal{D}(\mathbb{S}_2)$ in $\mathsf{Seq}^{(*)}$ (Theorem 5.14), and that $\mathsf{Seq}^{(*)}$-morphisms satisfy the condition $(\text{out})_{\mathsf{N}}$ (cf. page 1389). □

We can now show that for a step-by-step $\mathbb{S}$, the bisimilarity relations on $\mathcal{D}(\mathbb{S})$ and $\mathcal{N}(\mathbb{S})$ are identical.

Proposition 5.25. *Let $\mathbb{S}$ be a step-by-step subsequential structure. For all states q_1 and q_2 in $\mathbb{S}$, we have: $q_1 \sim_{\mathcal{S}_0^{(*)}} q_2$ in $\mathcal{D}(\mathbb{S})$ iff $q_1 \sim_{\mathcal{S}} q_2$ in $\mathcal{N}(\mathbb{S})$.*

Proof. Let $\mathbb{S} = (Q, o, d, r)$, $\mathcal{D}(\mathbb{S}) = (Q, \partial_{\mathbb{S}}, d)$ and $\mathcal{N}(\mathbb{S}) = (Q, o', d, r')$.

We will use the characterising conditions of $\mathcal{S}_0^{(*)}$-bisimilarity and $\mathcal{S}$-bisimilarity, (s0)–(s3) (page 1385) and (d0)–(d2) (page 1392), respectively.

Assume first that $q_1 \sim_{\mathcal{S}} q_2$ in $\mathcal{N}(\mathbb{S})$. Clearly, (d0) and (d2) follow from (s0) and (s2). For (d1) it suffices by Lemma 5.24 to show that $\partial_{\mathcal{N}(\mathbb{S})}(q_1)(a) = \partial_{\mathcal{N}(\mathbb{S})}(q_2)(a)$. By definition of $\mathcal{N}$ and ∂, we have for all $q \in Q$ and $a \in \text{supp}(q)$: $\partial_{\mathcal{N}(\mathbb{S})}(q)(a) = \overline{r'(q)} \centerdot o'(q)(a) \centerdot r'(d(q)(a))$. Now (d1) is easily seen to follow from (s1), (s2) and (s3).

Now assume that $q_1 \sim_{\mathcal{S}_0^{(*)}} q_2$ in $\mathcal{D}(\mathbb{S})$. Now (s0) and (s2) follow from (d0) and (d2). To see that (s3) holds, i.e., that $r'(q_1) = r'(q_2)$, suppose first that $\text{supp}(q_1) = \text{supp}(q_2) = \emptyset$. In this case, $\hat{\beta}(q_1) = r(q_1)$ and hence $r'(q_1) = \overline{\hat{\beta}(q_1)} \centerdot r(q_1) = \varepsilon$. Similarly, we get $r'(q_2) = \varepsilon$, and so $r'(q_1) = r'(q_2)$. Now suppose $\text{supp}(q_1) = \text{supp}(q_2) \neq \emptyset$. Since $\mathcal{N}(\mathbb{S})$ is normalised, there must be $a_1, a_2 \in \text{supp}(q_1)$ such that $lcp(\{r'(q_1), o'(q_1)(a_1)\}) = lcp(\{r'(q_2), o'(q_2)(a_2)\}) = \varepsilon$. From (d1) and Lemma 5.24 it follows that $\partial_{\mathcal{N}(\mathbb{S})}(q_1)(a_1) = \partial_{\mathcal{N}(\mathbb{S})}(q_2)(a_1)$, i.e.,

$$\overline{r'(q_1)} \centerdot o'(q_1)(a_1) \centerdot r'(d(q_1)(a_1)) = \overline{r'(q_2)} \centerdot o'(q_2)(a_1) \centerdot r'(d(q_2)(a_1)). \tag{16}$$

Letting $v = lcp(\{r'(q_2), o'(q_2)(a_1) \cdot r'(d(q_2)(a_1))\})$ it follows from the assumption on a_1 and (16) that $r'(q_2) = v \cdot r'(q_1)$. Using similar arguments, we get for a_2:

$$\begin{aligned} \overline{r'(q_1)} \cdot o'(q_1)(a_2) \cdot r'(d(q_1)(a_2)) &= \overline{r'(q_2)} \cdot o'(q_2)(a_2) \cdot r'(d(q_2)(a_2)) \\ &= \overline{r'(q_1)} \cdot \bar{v} \cdot o'(q_2)(a_2) \cdot r'(d(q_2)(a_2)). \end{aligned} \tag{17}$$

Since $v \preceq r'(q_2)$ we have by our choice of a_2 that $lcp(\{v, o'(q_2)(a_2)\}) = \varepsilon$. Hence from (17) we can now conclude that $v = \varepsilon$ and hence $r'(q_1) = r'(q_2)$.

It remains to prove (s1). First note that by the definition of $\mathcal{N}$, $r'(q) = r'(s)$ holds iff

$$\overline{\hat{\beta}(q)} \cdot r(q) = \overline{\hat{\beta}(s)} \cdot r(s) \quad \text{and hence} \quad \overline{\hat{\beta}(q)} = \overline{\hat{\beta}(s)} \cdot r(s) \cdot \overline{r(q)}.$$

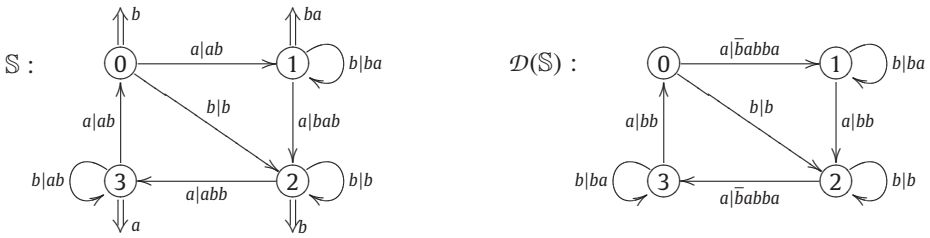
We now have,

$$\begin{aligned} & \overline{\hat{\beta}(q_1)} \cdot o(q_1)(a) \cdot \hat{\beta}(d(q_1)(a)) \\ (\text{s3}) = \ & \overline{\hat{\beta}(q_2)} \cdot r(q_2) \cdot \overline{r(q_1)} \cdot o(q_1)(a) \cdot r(d(q_1)(a)) \cdot \overline{r(d(q_2)(a))} \cdot \hat{\beta}(d(q_2)(a)) \\ (\text{d1}) = \ & \overline{\hat{\beta}(q_2)} \cdot r(q_2) \cdot \overline{r(q_2)} \cdot o(q_2)(a) \cdot r(d(q_2)(a)) \cdot \overline{r(d(q_2)(a))} \cdot \hat{\beta}(d(q_2)(a)) \\ = \ & \overline{\hat{\beta}(q_2)} \cdot o(q_2)(a) \cdot \hat{\beta}(d(q_2)(a)). \end{aligned}$$

Hence $o'(q_1)(a) = o'(q_2)(a)$. □

The next example illustrates the result of Proposition 5.25 and the difference between the two types of minimal realisations.

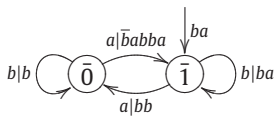
Example 5.26 (*minimal realisations*)**.** Consider the following transition diagram of a step-by-step subsequential structure $\mathbb{S} = (Q, o, d, r)$, and its differential representation $\mathcal{D}(\mathbb{S})$:



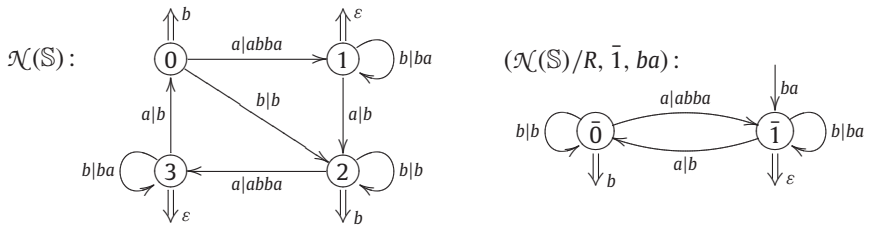
Let R be the least equivalence relation on S which contains $\{\langle 0, 2\rangle, \langle 1, 3\rangle\}$. It can easily be checked that R is the $\mathcal{S}_0^{(*)}$-bisimilarity relation on $\mathcal{D}(\mathbb{S})$. For $i \in \{0, 1, 2, 3\}$, let $\mathbb{T}_i = (\mathbb{S}, i, \varepsilon)$ and $f_i = [\![\mathbb{T}_i]\!] = [\![i]\!]_{\mathbb{S}}$.

From Theorem 5.23 and the observation that $r(0) = r(2)$ we can conclude that $f_0 = f_2$. For $i \in \{0, 1, 2, 3\}$, we can obtain a minimal sequential transducer with output in $B^{(*)}$ which realises f_i by quotienting $\mathcal{D}(\mathbb{S})$ with R and initialising this structure with the R-class containing i, adding initial prefix $f_i(\varepsilon) = r(i)$. Let $\bar{0} = \{0, 2\}$ and $\bar{1} = \{1, 2\}$. For f_1, this minimal realisation is:

$(\mathcal{D}(\mathbb{S})/R, \bar{1}, ba)$:



Alternatively, we could compute and minimise $\mathcal{N}(\mathbb{S})$. It can easily be verified that: $\hat{\beta}(0) = \varepsilon$, $\hat{\beta}(1) = ba$, $\hat{\beta}(2) = \varepsilon$, $\hat{\beta}(3) = a$. $\mathcal{N}(\mathbb{S})$ is illustrated below on the left. We now obtain a minimal normalised realisation of f_i by quotienting $\mathcal{N}(\mathbb{S})$ with R, initialising with the R-class containing i and adding the initial prefix $\hat{\beta}(i)$. For f_1, the minimal normalised realisation is shown below on the right:



We can now show that the minimisation of the differential representation is the differential representation of the minimisation.

Theorem 5.27. *Let $\mathbb{S} = (Q, o, d, r)$ be a step-by-step subsequential structure, and $\mathbb{T} = (\mathbb{S}, i, m)$ a step-by-step subsequential transducer. Let also*

$$\begin{aligned} Z &= \{\langle q_1, q_2\rangle \in Q \times Q \mid \text{in } \mathcal{D}(\mathbb{S}) : q_1 \sim_{S_0^{(*)}} q_2\} \\ &= \{\langle q_1, q_2\rangle \in Q \times Q \mid \text{in } \mathcal{N}(\mathbb{S}) : q_1 \sim_{\mathcal{S}} q_2\} \quad \textit{(cf. Proposition 5.25)}. \end{aligned}$$

We have:

$$\mathcal{D}(\mathcal{N}(\mathbb{S})/Z) \;=\; \mathcal{D}(\mathbb{S})/Z \quad \textit{and} \quad \mathcal{D}(\mathcal{N}(\mathbb{T})/Z) \;=\; \mathcal{D}(\mathbb{T})/Z.$$

Proof. We first show the result for a step-by-step structure $\mathbb{S} = (Q, o, d, r)$. Let $\mathcal{D}(\mathbb{S})/Z = (Q/Z, \partial_Z, d_Z)$. Since $\mathcal{D}$ and $\mathcal{N}$ do not change the underlying DA, the result follows once we show that $\partial_Z = \partial_{\mathcal{N}(\mathbb{S})/Z}$. Let $\rho : Q \to Q/Z$ be the quotient map which sends a state $q \in Q$ to its Z-class q/Z. Since $\rho : \mathcal{D}(\mathbb{S}) \to \mathcal{D}(\mathbb{S})/Z$ is a $\mathsf{Seq}^{(*)}$-morphism, and also $\rho : \mathcal{N}(\mathbb{S}) \to \mathcal{N}(\mathbb{S})/Z$ is a $\mathsf{Coalg}(\mathcal{S})$-morphism, we have for all $q \in Q$ and $a \in \mathrm{supp}(q)$:

$$\partial_{\mathbb{S}}(q)(a) = \partial_Z(q/Z)(a) \quad \text{and} \quad \partial_{\mathcal{N}(\mathbb{S})}(q)(a) = \partial_{\mathcal{N}(\mathbb{S})/Z}(q/Z)(a).$$

It follows now from Lemma 5.24 that $\partial_Z(q/Z)(a) = \partial_{\mathcal{N}(\mathbb{S})/Z}(q/Z)(a)$.

The proof for the transducer case follows from result for structures as soon as we can show that $\mathcal{D}(\mathbb{T})$ and $\mathcal{D}(\mathcal{N}(\mathbb{T}))$ have the same initial prefix. Let $\mathbb{T} = (Q, o, d, r, i, m) \in \mathsf{StepTra}$. The initial prefix of $\mathcal{D}(\mathbb{T})$ is $m \cdot r(i)$ (Definition 5.11). Letting m' and r' denote the initial prefix and the terminal output function in $\mathcal{N}(\mathbb{T})$, the initial prefix in $\mathcal{D}(\mathcal{N}(\mathbb{T}))$ is $m' \cdot r'(i) = m \cdot \hat{\beta}_{\mathbb{T}}(i) \cdot r'(i) = m \cdot r(i)$ (cf. Definition 3.25). □

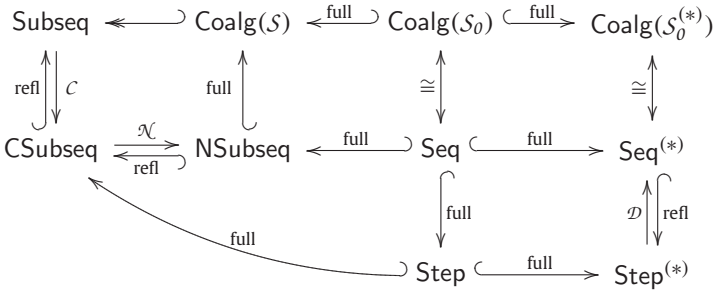
For the class of step-by-step structures, differential representations could offer an interesting alternative to normalised structures for the purpose of minimisation and deciding equivalence. We base this on the observation that computing $\hat{\beta}$ is a non-trivial task which involves a global fixpoint computation (cf. [4,10]) whereas differentials can be computed locally in the sense that for each state q and $a \in \mathrm{supp}(q)$, we only need to know the values of $r(q)$, $o(q)(a)$ and $r(d(q)(a))$ in order to determine $\partial_{\mathbb{S}}(q)(a)$. This local nature of the differential allows for flexible and straightforward algorithms for computing differential representations. It would be interesting to see how the two techniques compare in practice, and when applied to real examples.

6. Conclusion

Although subsequential structures as objects have the type of coalgebras for a functor $\mathcal{S} : \mathsf{Set} \to \mathsf{Set}$ their word function semantics requires a notion of morphism which is more general than the notion of $\mathcal{S}$-coalgebra morphism. Hence the category of all subsequential structures cannot be seen as a (sub)category of coalgebras. However, we showed that normalised structures and differential structures do have a coalgebraic modelling. Hence normalisation and taking differentials transform subsequential structures into an equivalent coalgebraic representation. This is what we mean when we say that normalisation and taking differentials are a form of coalgebraisation. One immediate consequence of this coalgebraisation is that finite structures can be minimised by quotienting with bisimilarity. We provided a detailed description of how one can adapt the known method for DFA-minimisation to normalised structures and differential representations. For the purpose of deciding equivalence of step-by-step transducers, we believe that the decision method obtained by computing state equivalence on the differential structure is an interesting alternative to the normalise–minimise method. This claim is based on the easy, local manner in which the differential can be computed, as opposed to the more complicated and global nature of known normalisation algorithms (cf. [4,10]).

The mathematical properties of the abovementioned transformations were made precise by showing that normalisation and taking differentials are reflectors $\mathcal{N}$ and $\mathcal{D}$, respectively. We can therefore argue that the right way of thinking about

subsequential structures is in their coalgebraic, normalised form, and for step-by-step structures, the differential representation yields an alternative, but equally correct coalgebraic description. In the diagram below we provide an overview of the relationships between the various classes of subsequential structures and coalgebras that have been studied in this paper. The inclusion arrows indicate embeddings of categories; a double-headed arrow indicates that the embedding is surjective on objects; and the labels 'full', 'refl' and $\cong$ indicate whether the embedding is full, reflective or an isomorphism, respectively.



As directions for future research, we mention that in automatic speech recognition, subsequential transducers with weighted transitions play an important role (cf. [30]). Since weighted or probabilistic systems can be modelled coalgebraically [3], one could try to extend the current coalgebraic modelling to weighted subsequential transducers. On the side of formal languages and transductions, the natural next step would be to try to give a coalgebraic modelling of sequential bimachines and rational functions [12]. We expect this to be a non-trivial exercise, if at all possible, since a unique canonical minimal bimachine seems not to exist [33].

The problems we encountered in the coalgebraic modelling of arbitrary subsequential structures arise from the presence of internal states and the fact that the word function semantics equates output in the free monoid. These issues could perhaps be dealt with by extending the coalgebraic setting from being purely set based, to one in which the monoid identities are formally included. Another idea would be to look for alternative equivalence notions along the lines of weak bisimilarity. However, weak bisimilarity in coalgebras is not very well understood, but some results may be found in [34,36,41]. In this context, we also mention that although subsequential structures seem to allow a kind of internal steps, these steps are not entirely unobservable since an input letter is always consumed.

Another direction for future research would be to find out whether existing coalgebraic specification languages (cf. [7,26, 31]) are useful for expressing properties in the application domains of subsequential transducers. For example, the regular expressions for polynomial coalgebras given in [7] provide an expressive formal language for specifying normalised and step-by-step subsequential transducers. We would also like to know if our results on step-by-step transducers and differential representations are practically useful in this or other application domains.

Finally, we mention that step-by-step transducers are implicitly used in [8] to give a proof of Choffrut's characterisation theorem for subsequential functions. It would be interesting to see if Choffrut's result can be reinterpreted in the coalgebraic setting presented here.

Acknowledgments

I am grateful to Jan Rutten and Clemens Kupke for their helpful suggestions and comments on earlier versions of this paper. Thanks are also due to Jean-Éric Pin for the discussions we had during his visit to Amsterdam in 2006, to Jose Proença for refreshing my memory about binary search trees, and to the anonymous referees whose comments helped me substantially improve the paper in its last stages.

References

[1] J. Adámek, H. Herrlich, G.E. Strecker, Abstract and Concrete Categories: The Joy of Cats, John Wiley and Sons, 1990. Online version: <http://katmat.math.uni-bremen.de/acc>
[2] J. Adámek, V. Trnková, Automata and Algebra in Categories, Kluwer Academic Publishers, 1990.
[3] F. Bartels, A. Sokolova, E.d. Vink, A hierarchy of probabilistic system types, Theoretical Computer Science 327 (2004) 3–22.
[4] M.-P. Béal, O. Carton, Computing the prefix of an automaton, Theoretical Informatics and Applications (RAIRO) 34 (2000) 503–514.
[5] M.-P. Béal, O. Carton, Determinization of transducers over finite and infinite words, Theoretical Computer Science 289 (2002) 225–251.
[6] J. Berstel, Transductions and Context-Free Languages, Leitfäden der angewandten Mathematik und Mechaniek, vol. 38, B.G. Teubner, Stuttgart, 1979.
[7] M.M. Bonsangue, J.J.M.M. Rutten, A.M. Silva, An algebra for Kripke polynomial coalgebras, in: Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science (LICS 2009), 2009.
[8] V. Bruyère, C. Reutenauer, A proof of Choffrut's theorem on subsequential functions, Theoretical Computer Science 215 (1999) 329–335.
[9] C. Choffrut, A generalization of Ginsburg and Rose's characterization of g-s-m mappings, in: H.A. Maurer (Ed.), Proceedings of the 6th International Colloquium on Automata, Languages and Programming (ICALP 1979), Lecture Notes in Computer Science, vol. 71, 1979, pp. 88–103.
[10] C. Choffrut, Minimizing subsequential transducers: a survey, Theoretical Computer Science 292 (2003) 131–143.
[11] J.H. Conway, Regular Algebra and Finite Machines, Mathematics Series, Chapman and Hall, 1971.

[12] S. Eilenberg, Automata, Languages and Machines, vol. A, Academic Press, 1974.
[13] C. Frougny, Numeration systems, in: M. Lothaire (Ed.), Algebraic Combinatorics on Words, Cambridge University Press, 2002 (Chapter 7).
[14] D. Gries, Describing an algorithm by Hopcroft, Acta Informatica 2 (1973) 97–109.
[15] H.P. Gumm, Functors for coalgebras, Algebra Universalis 45 (2001) 135–147.
[16] H.P. Gumm, On minimal coalgebras, Applied Categorical Structures 16 (2008) 313–332.
[17] H.P. Gumm, T. Schröder, Types and coalgebraic structure, Algebra Universalis 53 (2005) 229–252.
[18] H.H. Hansen, Coalgebraising subsequential transducers, in: Proceedings of the 9th Workshop on Coalgebraic Methods in Computer Science (CMCS 2008), Electronic Notes in Theoretical Computer Science, vol. 203(5), 2006, pp. 109–129.
[19] J.E. Hopcroft, An $n \log n$ algorithm for minimizing states in a finite automaton, in: Z. Kohavi, Z. Paz (Eds.), Theory of Machines and Computation, Academic Press, 1921, pp. 189–196.
[20] J.E. Hopcroft, R. Motwani, J.D. Ullman, Introduction to Automata Theory, Languages and Computation, second ed., Addison-Wesley, 2001.
[21] D. Jurafsky, J.H. Martin, Speech and Language Processing, second ed., Prentice Hall, 2008.
[22] D.E. Knuth, The Art of Computer Programming, third ed., Sorting and Searching, vol. 3, Addison-Wesley, 1997.
[23] T. Knuutila, Re-describing an algorithm by Hopcroft, Theoretical Computer Science 250 (2001) 333–363.
[24] D. Kozen, Automata and Computability, Undergraduate Texts in Computer Science, Springer, 1997.
[25] D. Kozen, On the coalgebraic theory of Kleene algebra with tests, Technical Report URI: <http://hdl.handle.net/1813/10173>, Computing and Information Science, Cornell University, 2008.
[26] C. Kupke, Y. Venema, Coalgebraic automata theory: basic results, Logical Methods in Computer Science 4 (2008).
[27] S. Mac Lane, Categories for the Working Mathematician, Graduate Texts in Mathematics, vol. 5, Springer, 1998.
[28] G.H. Mealy, A method for synthesizing sequential circuits, Bell System Technical Journal 34 (1955) 1045–1079.
[29] M. Mohri, Finite-state-transducers in language and speech processing, Computational Linguistics 23 (1997) 269–311.
[30] M. Mohri, F.C.N. Pereira, M. Riley, Speech recognition with weighted finite-state transducers, in: L. Rabiner, F. Juang (Eds.), Handbook of Speech Processing and Speech Communication, Part E: Speech Recognition, Springer-Verlag, 2008.
[31] D. Pattinson, Coalgebraic modal logic: soundness, completeness and decidability of local consequence, Theoretical Computer Science 309 (2003) 177–193.
[32] C. Reutenauer, Subsequential functions: characterizations, minimization, examples, in: Aspects and Prospects of Theoretical Computer Science, Proceedings of the 6th International Meeting of Young Computer Scientists (IMYCS 1990), LNCS, vol. 464, 1990, pp. 62–79.
[33] C. Reutenauer, M.-P. Schützenberger, Minimization of rational word functions, SIAM Journal of Computing 20 (1991) 669–685.
[34] J. Rothe, D. Mašulović, Towards weak bisimulation for coalgebras, in: A. Kurz (Ed.), Categorical Methods for Concurrency, Interaction, and Mobility, Electronic Notes in Theoretical Computer Science, vol. 68(1), 2002, pp. 32–46.
[35] J.J.M.M. Rutten, Automata and coinduction (an exercise in coalgebra), in: D. Sangiorgi, R. de Simone (Eds.), Proceedings of 9th International Conference on Concurrency Theory (CONCUR 1998), Lecture Notes in Computer Science, vol. 1466, 1998, pp. 194–218.
[36] J.J.M.M. Rutten, A note on coinduction and weak bisimilarity for while programs, Technical Report SEN-R9826, Centrum voor Wiskunde en Informatica (CWI), 1998.
[37] J.J.M.M. Rutten, Universal coalgebra: a theory of systems, Theoretical Computer Science 249 (2000) 3–80.
[38] J.J.M.M. Rutten, Behavioural differential equations: a coinductive calculus of streams, automata and power series, Theoretical Computer Science 308 (2003) 1–53.
[39] J.J.M.M. Rutten, Algebraic specification and coalgebraic synthesis of Mealy machines, in: Proceedings of the 2nd International Workshop on Formal Aspects of Component Software (FACS 2005), Electronic Notes in Theoretical Computer Science, vol. 160, 2005, pp. 305–319.
[40] M. Schützenberger, Sur un variante des fonctions sequentielles, Theoretical Computer Science 4 (1977) 47–57.
[41] A. Sokolova, E. de Vink, H. Woracek, Weak bisimulation for action-type coalgebras, in: L. Birkedal (Ed.), Proceedings of Category Theory and Computer Science (CTCS 2004), Electronic Notes in Theoretical Computer Science, vol. 122, 2005.